
الأساسـيات في الهندسة المعلوماتية
Essentials in Informatics Engineering


إعداد
الدكتور باسل الخطيب
الدكتور خليل عجمي
الدكتور راكان رزوق
الدكتور سعيد أبو تراب
الدكتورة سيرا استورا
الدكتور عمار جوخدار
الدكتور غسان سايا
الدكتورة مادلين عبود
الدكتور محمد جنيدي
الدكتور مروان زبيبي
الإشراف والتنسيق العام
الدكتورة ميسون دشاش

2018

بسم الله الرحمن الرحيم

لما غدا تأمين الكتاب المرجعي مطلباً ملحاً وحاجة أساسية للطلاب الدارسين في الكليات الهندسية بالجامعات السورية، ورغبة من مركز المنار للكتاب في تسهيل الطريق لأبنائنا الطلاب بالوصول الى المصادر الموثوقة بأسهل الطرائق وأوفرها، كان لزاماً العمل جاهدين لتبني كتاب «**الأساسيات في الهندسة المعلوماتية**» واستحصال موافقات نشره كجزء من مجموعة أعمال قدمها مركز المنار لدعم مشروع مركز القياس والتقويم في ترجمة وإعداد وتأليف الكتب المرجعية لطلاب الجامعات السورية.

حيث كانت باكورة الأعمال كتاب «**الواضح في طب الأطفال**»، ويُعد مشروع ترجمة وإعداد وتأليف الكتب التخصصية مشروعاً رائداً بين القطاع الخاص الوطني والقطاع الحكومي المخلص حيث يهدف إلى ترجمة وإعداد عدد من العناوين الأهم في العالم في اختصاصات الطب البشري وطب الأسنان والصيدلة والمعلوماتية والهندسة المعمارية وغيرها ووضعها بين يدي الطالب السوري.

وإن هذا المشروع يهدف في استراتيجيته المعرفية إلى جعل الطالب السوري مواكباً للإنتاجات العالمية العلمية الاختصاصية الحديثة والمعاصرة من خلال تقديم أحدث العناوين المؤلفة والمترجمة من قبل أشهر المؤلفين العالميين والوطنيين ووضعها بين يديه الأمر الذي سيمكنه من السبق في الحصول على المادة العلمية الأحدث في العالم.

إن تأليف وإعداد وترجمة هذه العناوين الحديثة والمتميزة على مستوى العالم باللغة العربية مع توفر بنك الأسئلة المتعلق بالمادة يُعد قفزة نوعية في القياس والتقويم الموضوعي للعملية التعليمية والمادة التدريسية ولبنة اساسية في استمرار التقدم والتطور في مستوى الكتاب المرجعي الخاص بالامتحان الوطني السوري مما يمكّن الادارة العليا من اتخاذ القرارات الصحيحة والمناسبة في هذا الخصوص. هذا وسيتم تحديث هذه العناوين دورياً وكما سيتم تقديم كافة مصادر المدرس للأساتذة المدرسين لتمكينهم من الاستفادة من التحديثات الدورية للمادة العلمية.

إن التعاون والعمل المتكامل بين مركز المنار للكتاب ومركز القياس والتقويم يُعد الأساس في إنجاح هذه التجربة الرائدة ولابد لنا من الإشادة الكبيرة بالعمل الدؤوب والمخلص والمتابعة الحثيثة التي أبدتها **الدكتورة ميسون دشاش** مدير عام مركز القياس والتقويم في التعليم العالي لجعل هذا المشروع عملاً متميزاً بكل المقاييس في خدمة الطلاب والتعليم العالي.

وإن مركز المنار للكتاب يتقدم بالشكر الجزيل الى كل من ساهم في انجاح هذا العمل وإخراجه إلى النور كما يرجو ويأمل أملاً كبيراً أن يكون في هذا المرجع وغيره من المراجع اللاحقة كل الخير والفائدة العلمية والعملية لكل من نذر نفسه لخدمة رسالة العلم والسعي الدؤوب إلى تطويرها في وطننا العزيز

والله ولي التوفيق

دمشق في *16* تشرين الأول *2018*

الناشر

تقديم

استطاع التقدم التقني والمعلوماتي رفع الحواجز وتقريب المسافات إلى حد جعل العالم قرية صغيرة تتداخل فيها الأفكار والعلوم فلا مجال لنصب أسوار العزلة، إذ أصبح الفرز واضحاً بين من يبتكر التقانة ويستثمرها وبين من يستهلكها وبمهارات محدودة.

ألقت نواتج المعلوماتية بظلالها علينا فلابد من إعداد وتأهيل جيل قادر على مواكبة كل ما هو جديد وبسرعة حتى لا يكون مستهلكا للتقدم التقني وإنما منتجاً ومبتكراً، ولذلك غدا كتاب «الأساسيات في الهندسة المعلوماتية» حاجة ماسة لخريجي كليات المعلوماتية في الجامعات السورية قام بإعداده مركز القياس والتقويم في التعليم العالي كخطوة على درب تعليم العلوم الطبية ليضع اللبنة الأولى على درب ضمان كفاءة مخرجات التعليم العالي في مجال الهندسة المعلوماتية.

شارك في إعداد هذا الكتاب صفوة من خيرة الأساتذة في الجامعات السورية كمحاولة لإثراء مكتبتنا العربية بأحدث المعلومات والمعرفة ليكون مرشداً للدارس والمدرس وحافزاً للتعلم الذاتي واكتساب المهارات.

يُطّل هذا الكتاب علينا بمعلومات حديثة في مجال الخوارزميات وقواعد البيانات والبرمجيات حيث يشرح الأوتومات واللغات الصورية والمترجمات بلغة سهلة وواضحة، ويشرح هندسة البرمجيات وأساسيات الذكاء الصنعي والنظم الخبيرة بطريقة مبسطة وكذلك يفصل في باب الشبكات الحاسوبية ويشرح أساسيات أمن النظم المعلوماتية ومهددات النظم المعلوماتية وأمن الشبكات وأساسيات العلوم الحديثة كالجرائم الالكترونية وعلوم الويب والانترنت.

تضمن الكتاب أهم المواضيع في الهندسة المعلوماتية وأدرجت مخططات ورسوم توضيحية تساعد في وصول المعلومة بطريقة سهلة وواضحة. كما تضمن الكتاب تمارين ونماذج امتحانية تساعد الطلاب في المراجعة والإعداد للاختبارات الوطنية الموحدة.

وإنني باسم مركز القياس والتقويم أتوجه بالشكر الجزيل لكل من ساهم في إنجاز هذا العمل الجبار وأخص الأساتذة الذين تطوعوا بجهدهم ووقتهم لإنجاز هذا العمل وهم الدكتور باسل خطيب والدكتور خليل عجمي والدكتور راكان رزوق والدكتور سعيد أبو تراب والدكتورة سيرا استورا والدكتور عمار جوخدار والدكتور غسان سابا والدكتورة مادلين عبود والدكتور محمد جنيدي والدكتور مروان زبيبي والدكتور خالد العمر وأشكر فريق مركز القياس والتقويم الذي شاركني في تدقيق الكتاب وإخراجه بالصورة الحالية وخلال فترة وجيزة، وفريق مركز المنار للكتاب الذي ساهم في تقديم الدعم المالي والتقني لتغطية نفقات الأساتذة ونفقات الإخراج والنشر وتابع العمل مع المركز حتى يرى الكتاب النور.

أرجو من الله أن يحقق الكتاب الهدف المرجو من إعداده وأن يعود بالنفع على بلدنا الحبيب سورية.

المنسق العام الدكتورة ميسون دشاش

مدير عام مركز القياس والتقويم في التعليم العالي



المقدمة

يفرض التطور المطّرد - لما بات يُعرف اليوم باقتصاد المعرفة القائم على التعليم والبحث العلمي - تحدياً وجودياً يدفع دول العالم لتطوير منظومات التعليم فيها والاستثمار بها لتأهيل شرائح كبيرة من البشر وتسليحهم بالمعرفة الضرورية للمنافسة في سوق العمل المعولم.

وتشكل الجامعات والمؤسسات العلمية إضافةً إلى هيئات الاعتمادية الأكاديمية الركيزة الأساسية لعملية التطوير تلك كونها مرجعيات علمية ومعنوية ومهنية واجتماعية وأخلاقية، مسؤولةً عن رسم سياساتٍ طويلة الأمد وفق تخطيطٍ استراتيجيٍ مدروس، يضمن جودة التعليم ومصداقيته على المستوى الوطني.

استناداً لما سبق، تأتي الامتحانات الوطنية الموحدة كخطوةٍ متقدمة تخطوها المؤسسات الأكاديمية السورية نحو تعميق الثقة بمخرجات العملية التعليمية فيها وترسيخ اعتماديتها، وهي خطوةٌ لا يمكن النظر إليها بمعزلٍ عن مجموعةٍ من إجراءات الاعتمادية الأخرى التي ستضمن لخريج الجامعات السورية أن يتحقق قبل غيره من جودة الخدمة الأكاديمية التي حصل عليها من مؤسسته التعليمية.

وكخطوةٍ متواضعةٍ في هذا الاتجاه، نضع كتابنا هذا بين أيدي طلابنا الأعزاء، لنقدم من خلاله دليلاً توجيهياً للمجالات المعرفية الرئيسية في اختصاص الهندسة المعلوماتية، والتي يتوجب على الطالب تحصيلها خلال دراسته الأكاديمية لمختلف التخصصات الدقيقة. فضمان هذا التحصيل بات يُعتبر أمراً حيوياً في سعينا لترسيخ اعتمادية الشهادة الجامعية السورية التي نفخر ونعتز بها وترسيخ مكانتها على المستويين الإقليمي والدولي، آملين أن نكون قد وفقنا في تحقيق ما نسعى إليه خدمةً لوطننا الحبيب سورية.

رئيس الجامعة الافتراضية

الدكتور خليل عجمي باسم المؤلفين



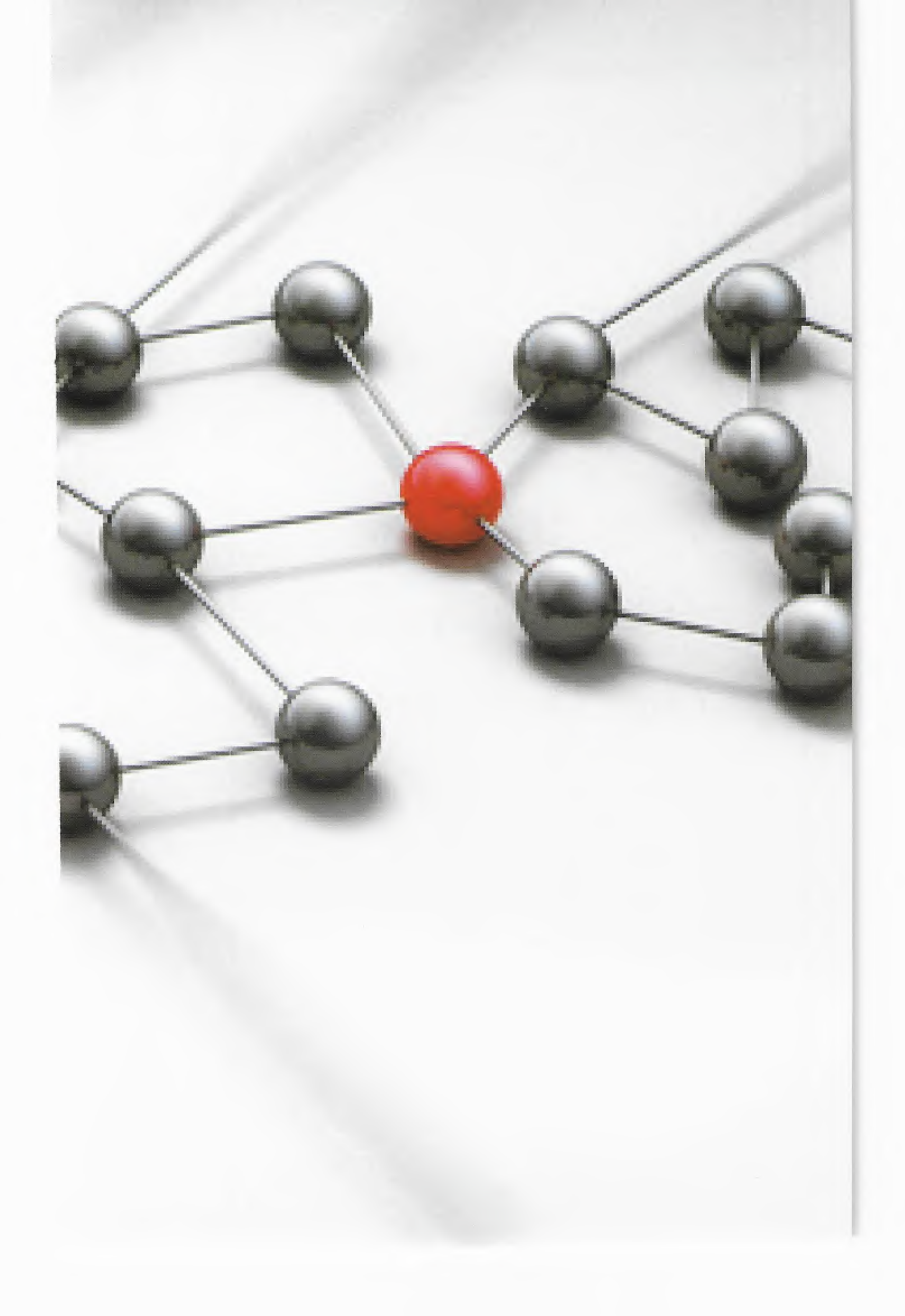
الإشراف والتنسيق العام الدكتورة ميسون دشاش	
فريق الإعداد	
الدكتور باسل الخطيب	الدكتور خليل عجمي
الدكتور راكان رزوق	الدكتور سعيد أبو تراب
الدكتورة سيرا استورا	الدكتور عمار جوخدار
الدكتور غسان سابا	الدكتورة مادلين عبود
الدكتور محمد جنيدي	الدكتور مروان زيبي
الدكتور خالد عمر	
الإخراج الطباعي	
الأستاذ محمد نظام	تسنيم الشلبي
نضال بسام ناعورة	عبد المنعم حسين


http:/www.

الباب الأول

الخوارزميات وقواعد البيانات

(Algorithms and Databases)



الفصل الأول

مفاهيم أساسية في الخوارزميات

1 - 1- تعاريف

خوارزمية ***(Algorithm)*** حلّ مسألة، هي توصيف صوري لطريقة الحل على شكل متتالية منتهية من العمليات البسيطة، تنفَّذ حسب تسلسل محدّد.

البرنامج ***(Program)*** هو توصيف لخوارزميّة حلّ مسألة معيّنة بإحدى لغات البرمجة التي يقبلها الحاسوب.

لغة البرمجة ***(Programming Language)*** هي مجموعة من المفردات والقواعد والدلالات المعرّفة التي تسمح بكتابة برنامج يمكن تنفيذه على الحاسوب.

المترجم ***(Compiler)*** هو برنامج يفهم البرنامج المكتوب بلغة برمجة معينة، ويحوله إلى برنامج مكافئ مكتوب بلغة المجمع *(Assembly Language)* الخاصة (أو أحياناً لغة الآلة نفسها) بالمعالج الصغري *(Microprocessor)* للحاسوب.

كي تصبح الخوارزميّة قابلة للتنفيذ على الحاسوب يجب ترجمتها إلى تعليمات وفق لغة برمجة متوفّرة في الحاسوب.

الخوارزميّة مستقلة عن لغة البرمجة المستخدمة، فخوارزميّة إقليدس لإيجاد القاسم المشترك الأعظم لعددين، تبقى نفسها بأي لغة كُتبت سواء بـ *PASCAL* أو *ADA* أو *LISP*.

1 - 2 - أمثلة لخوارزميات بسيطة

1 - البحث التسلسلي عن عنصر ضمن قائمة

{يقوم هذا البرنامج بالبحث عن موضع العنصر X في القائمة L}

{المعطيات هي L و X.}

{القائمة L تحوي n عنصراً ويمكن لهذه القائمة أن تكون فارغة

$(n = 0)$.}

{تبحث الخوارزميّة عن دليل العنصر X ضمن القائمة L.}

{عندما تتوقّف الخوارزميّة يحوي المتحوّل j دليل أوّل ظهور لـ X في L.}

{وإذا كان X غير موجود في L فإن j يحوي القيمة صفر.}

```
j := 1;
while ( j <= n ) and ( L[j] <> X ) do
        j:= j+1;
if j > n  then
        j:= 0;
```

2- مسألة الفرز بالدمج

ليكن لدينا الجدول T، عناصره من الأعداد الصحيحة غير المرتبة، ونريد ترتيب هذه العناصر ضمن الجدول بطريقة الفرز بالدمج *(Merge Sort)*. وتتلخص هذه الطريقة بما يلي: نقسّم الجدول إلى جدولين جزئيين، ونرتب كل جدول جزئي على حدة، ثمّ نقوم بدمجهما في جدول مرتب واحد.

عملية ترتيب كل جدول جزئي هي تطبيق لنفس الطريقة لكن عند مستوى أقل. نتابع هذه العملية وصولاً إلى جداول جزئية من مرتبة خانتين فقط. فهذه يمكن ترتيبها بسهولة.

نحن إذن أمام مسألة عوديّة. نلجأ في حلها إلى الخطوات التالية:

1 - بنى المعطيات المستخدمة

نعرّف الجدول بواسطة تسجيلة *(Record)*: يدّل الحقل الأول على عدد الخانات المملوءة فعلاً (بُعد الجدول). أمّا الحقل الثاني فيحوي الجدول نفسه.

```
const
        n = 100;
type
        index = 1..n;
        TableType = record
        dim : index;
        Table : array[index] of integer
        end;
```

2 - الخوارزمية

نريد إذن كتابة خوارزمية الفرز بالدمج.

الترويسة

procedure MergeSort (low, high : index; var T : TableType);

الدخل

جدول *T* من الأعداد الصحيحة غير المرتبة. أدلة الحدّ الأدنى والحدّ الأعلى للجدول.

الخرج

نفس الجدول *T* لكن مرتباً تصاعدياً.

مخطط الخوارزمية

1- مادام شرط التوقف غير محقق

1-2- احسب دليل منتصف الجدول *T* وضع القيمة في المتحول *mid*.

1-2- رتّب الجدول الجزئي *T[1..mid]* بطريقة الفرز بالدمج.

1-3- رتّب الجدول الجزئي *T[mid+1..T.dim]* بطريقة الفرز بالدمج.

1-4- ادمج الجدولين الجزئيين المرتبين في جدول وحيد مرتب *T[1..T.dim]*.

ترميز الخوارزمية

لسهولة مخطط الخوارزمية لا داعي لكتابة ترميز لكل مرحلة. لذلك نجد فيما يلي الترميز الكامل للخوارزمية:

```
procedure MergeSort (low, high : index; var T : TableType);
var
        mid : 1..n;
begin
        if low <> high then  { Stop Condition }
        begin
        mid := (low + high) div 2;
        MergeSort(low, mid, T);
        MergeSort(mid + 1, high, T);
        Merge(low, mid, high, T)
        end
end;
```

لاحظ أن الاستدعاء الأولي لهذه الإجرائية في جسم البرنامج سيكون *MergeSort(1,T.mid,T)*. نلاحظ أننا استخدمنا في هذه الخوارزمية الإجرائية الجزئية *Merge* لدمج جدولين جزئيين مرتبين في جدول مرتب وحيد. هذه الإجرائية الجزئية تحتاج أيضاً إلى توصيف.

خوارزمية الدمج (*Merge*)

ترويسة الإجرائية

procedure Merge(low, mid, high : index; var T : TableType);

الدخل

أدلة بداية ونهاية الجدولين الجزئيين في الجدول T والجدول T.

الخرج

الجدول T مرتب.

ملاحظة

نفترض أن الجدولين الجزئيين متجاوران، بمعنى أن دليل بداية الجدول الجزئي الثاني تساوي دليل نهاية الجدول الجزئي الأول $+1$.

مخطط الخوارزمية

نستخدم جدولاً مساعداً Aux لتخزين العناصر قبل نقلها إلى الجدول T.

نستخدم أيضاً ثلاثة عدّادت:

i يمسح الجدول الجزئي الأول (نسميه هنا $T1$ للتبسيط).

j يمسح الجدول الجزئي الثاني (نسميه هنا $T2$ للتبسيط).

k يمسح الجدول المساعد.

1- ما دام لم ينته أحد الجدولين على الأقل:

1-1- إذا كان $T1[i] < T2[j]$

ننسخ $T1[i]$ في الخانة $Aux[k]$

نزيد العدّاد i واحداً، ونزيد العداد k واحداً.

1-2- إذا كان $T1[i] > T2[j]$

ننسخ $T1[j]$ في الخانة $Aux[k]$

نزيد العدّاد j واحداً، ونزيد العداد k واحداً.

1-3- إذا كان $T2[j] = T1[i]$

ننسخ $T1[i]$ في الخانة $Aux[k]$ ، وننسخ $T2[j]$ في الخانة $Aux[k+1]$

نزيد العدّاد i واحداً، والعدّاد j واحداً، والعدّاد k اثنين.

2- إذا انتهى الجدول $T1$ قبل الجدول $T2$

ننسخ ما تبقى من الجدول $T2$ في الجدول Aux

3- إذا انتهى الجدول $T2$ قبل الجدول $T1$

ننسخ ما تبقى من الجدول $T1$ في الجدول Aux

4- ننسخ الجدول Aux في الجدول T.

ترميز الخوارزمية

لسهولة مخطط الخوارزمية لا داعي لكتابة ترميز لكل مرحلة. لذلك نجد فيما يلي الترميز الكامل للخوارزمية:

```
procedure Merge(low, mid, high : index; var T : TableType);
var
        i, j, k : index;
begin
        i := 1; k := 1;
        j := mid + 1;
        while (i <= mid) and (j <= high) do

        if T[i] < T[j] do

        begin

        aux[k] := T[i];

        k := k + 1; i := i + 1

        end

        else if T[i] > T[j] then

        begin

        aux[k] := T[j];

        k := k + 1; j := j + 1

        end

        else
        begin

        aux[k]   := T[i];

        aux[k+1] := T[j];

        k := k + 2;
```

```
        j := j + 1; i := i + 1;
        end;
        while j <= high do
        begin
        aux[k] := T[j];
        k := k + 1; j := j + 1
        end;
        while i <= mid do
        begin
        aux[k] := T[i];
        k := k + 1; i := i + 1
        end;
        i := 1;
        while i <= high do
        begin
        T[i] := aux[i];
        i := i + 1
        end;
end;
```

1 - 3 - تمارين غير محلولة

1- نريد حساب وطباعة مثلث باسكال من السطر 1 حتى السطر n باستخدام سلسلة خطية واحدة فقط (نمثل فيها العناصر المختلفة عن الواحد فقط). بعد حساب السطر i فإن السلسلة الخطية تحوي فقط عناصر السطر i (المختلفة عن الواحد). من الطبيعي أن حساب عناصر السطر i يعتمد على عناصر السطر $i - 1$ وفق القاعدة المعروفة:

$t[i, j] = t[i - 1, j] + t[i - 1, j - 1]$ حيث j هو موقع العنصر في السطر i من المثلث t.

اقترح بنى المعطيات المناسبة (دعِّم شرحك برسم توضيحي).

اكتب إجرائية تقوم بحساب عناصر السطر i وتخزينه في السلسلة الخطية (لاحظ أن نفس السلسلة تحوي عناصر السطر السابق $i-1$ قبل استدعاء هذه الإجرائية).

اكتب إجرائية تقوم بطباعة أول n سطراً من مثلث باسكال وفق الشكل النظامي (انظر الرسم التوضيحي الذي يحوي ستة الأسطر الأولى من المثلث).

```
1
1  1
1  2  1
1  3  3  1
1  4  6  4  1
1  5  10 10 5  1
```

2- اكتب بلغة الخوارزميات (*Pseudo Code*) خوارزمية تقرأ حرفاً واحداً من الأحرف المستخدمة في نظام العد الست عشري *Hexadecimal* وتحسب مقابل هذا الحرف في النظام العشري (الأحرف المستخدمة في النظام الست عشري هي *A B C D E F 9 8 7 6 5 4 3 2 1 0*). وناقش حالة الحروف التي لا تنتمي إلى حروف النظام الست عشري

3- حول الخوارزمية السابقة إلى إجرائية دخلها حرف وخرجها متحولان الأول منطقي يبين إذا كان الحرف المعطى في الدخل هو أحد حروف النظام الست عشري والثاني عدد صحيح يعبر عن قيمة الحرف المُدخَل في النظام العشري.

4- استخدم الإجرائية السابقة في برنامج يقرأ سلسلة من الحروف تعبر عن عدد ممثَّل في النظام الست عشري ويحسب قيمة هذا العدد في النظام العشري ويعطي رسالة خطأ إذا كان أحد الحروف المُدخَلة لا ينتمي إلى حروف النظام الست عشري.

5- يتضمن البرنامج التالي أخطاء كثيرة. أوجد خمسة منها وبين نوع الخطأ واكتب التعليمة الصحيحة.

```
Program Full-Of_Errors;
Var x,y,z :reel
i: integer,  b : Boolean;
Begin  Readln (x,y,z,b); i :=0;
         For k:= 1 to z do

         i:= i+y+k; b:= (i>x);
         writeln (b)
end.
```

6- اكتب تابعاً عودياً يقوم بحساب التابع التالي (n و m عددان طبيعيان) :

$A(0,m) = m+1 \qquad m > 0$

$A(n,0) = A(n-1, A(n-1,1)) \qquad n > 0$

$A(n,m) = A(n-1, A(n-1,m-1) \qquad n, m > 0$



الفصل الثاني

تعقيد الخوارزميات

2 - 1- المقدمة

ولدت دراسة تعقيد الخوارزميات مع استخدام الحواسيب ذات الأداء المتزايد باطراد لتنفيذ التعليمات الصريحة وغير الغامضة للخوارزمية التي تحل مسألة.

نستطيع من الناحية النظرية اللجوء إلى الحاسوب من أجل تنفيذ *(Execution)* كل مسألة محلولة بواسطة خوارزمية صحيحة. على هذا سيكون من المدهش وجود مسائل لا يمكن تنفيذها على الحاسوب (برغم إمكان حلها بواسطة خوارزميات صحيحة)، لأن تنفيذ هذه الخوارزميات قد يستغرق مئات السنين على أكثر الحواسيب سرعةً.

إذن يكتسب موضوع فعالية الخوارزميات (وحساب زمن تنفيذها) من الناحية العملية أهمية بديهية. لذلك تقوم النظريات الحالية في الحوسبة والتعقيد على طبيعة الحسابات وزمن تنفيذها من أجل توضيح كيفية اعتبار بعض المسائل أكثر صعوبة في التنفيذ من غيرها.

يتطلّب تنفيذ خوارزميّة على حاسوب معيّن استخدام الموارد التي يتيحها هذا الحاسوب. ونقصد بالموارد: المعالج (حجز المعالج لمدة زمنية معيّنة) والذاكرة المركزية والمحيطيّات المتّصلة بالحاسوب. سنركّز اهتمامنا أثناء دراسة تعقيد الخوارزميّات في عاملين أساسيين:

- **زمن التنفيذ:** ونقصد به الزمن اللازم لتنتهي الخوارزميّة من إنجاز تعليماتها كافّة. يقاس زمن التنفيذ بحساب عدد التعليمات والزمن اللازم لتنفيذ كل تعليمة.
- **حجم الذاكرة:** اللازمة لتخزين البرنامج والمعطيات التي يعالجها.

إن الهدف من تحليل الخوارزميّات لا يقف عند قياس المقدارين السابقين، بل يفيد في مقارنة خوارزميّات حل مسألة معيّنة. فما نريد الوصول إليه هو حكم من الطراز:

"مهما تكن الآلة التي ستنفّذ الخوارزميّة ومهما تكن لغة البرمجة المستخدمة فإن الخوارزميّة A أفضل من الخوارزميّة B من أجل معطيات ذات حجم كبير"

أو من الطراز:

"إن الخوارزميّة A هي مُثلى من حيث عدد العمليات الأساسية (أو المكلفة) التي تقوم بها لحل المسألة Q"

2 - 2 - حساب زمن تنفيذ برنامج

لتكن الخوارزمية A التي تحل المسألة P. إن كتابة برنامج تعني تمثيل A في النموذج الحسابي المعرّف بواسطة لغة عالية المستوى (مثل *Pascal* أو *C* أو *ADA*).

يعتمد زمن تنفيذ البرنامج على حاسوب على عوامل عديدة منها:

- معطيات المسألة P الخاصة بتجربة ما.
- جودة الرِماز *(Code)* الذي يولده المترجم من أجل بناء الملف التنفيذي.
- طبيعة وسرعة التعليمات المتوفرة في الحاسوب (في المعالج الصغري).

- جودة البرنامج الذي يكتبه المبرمج.
- فعالية الخوارزمية A.

من الملاحظ أن معظم هذه العوامل (عدا الأخير) خاصة: معطيات خاصة، حاسوب خاص، مقدرة المبرمج. لذلك من الطبيعي عدم اللجوء إلى مقارنة البرامج، والاهتمام بمقارنة الخوارزميات التي تحل نفس المسألة (أو اختيار الخوارزمية المثلى التي تحل مسألة ما من صف الخوارزميات التي تحل هذه المسألة). نبحث إذن عن مقياس يعكس جودة الخوارزميات بغض النظر عن تفاصيل تحويلها إلى برامج.

لحساب زمن تنفيذ خوارزميّة يجري التركيز في البداية في مسألتين أساسيتين:

الأولى: تحديد بُعد (أو طول أو حجم) معطيات المسألة من أجل كتابة درجة التعقيد بدلالة هذا البعد.

أمثلة

- **في مسائل كثيرات الحدود *(smonyloP)*:** يكون البعد هو درجة كثير الحدود أو عدد الأمثال.
- **في مسائل المصفوفات *(secirtaM)* من المرتبة *(n×m)*:** يكون البعد بدلالة أبعاد المصفوفة مثل *(n ,m)xam* أو $n+m$ أو $n.m$.
- **في مسائل البيانات *(shparG)*:** يكون البعد بدلالة عدد العقد *(secitreV)* أو عدد الأسهم *(segdE)* أو مجموعها.
- **في مسائل الفرز أو الترتيب *(gnitroS)*:** يكون البعد بدلالة عدد العناصر التي نريد ترتيبها.
- **في مسائل التحليل القواعدي *(sisylanA xatnyS)*:** يكون البعد بدلالة طول الكلمة.



الثانية: اختيار نوع أو عدّة أنواع من العمليات، بحيث يتناسب زمن تنفيذ الخوارزمية مع عدد هذه العمليات. يعتمد هذا الخيار على زمن تنفيذ هذه العمليات، إذ تُهمل العمليات البسيطة أمام العمليات التي هي أكثر كلفة.

أمثلة

- في خوارزميّة البحث عن عنصر ضمن سلسلة يجري التركيز في عدد عمليات المقارنة (وهي أكثر كلفة في هذه الحالة) بين العنصر الذي نبحث عنه وعناصر السلسلة.
- في خوارزمية فرز (ترتيب) العناصر في سلسلة يجري التركيز في عدد عمليات المقارنة بين العناصر وعدد عمليات التبديل بين مواقع العناصر.
- في خوارزميّة ضرب مصفوفتين يجري التركيز في عدد عمليات ضرب وجمع الأعداد.

بعد تحديد أنواع العمليات الأساسية لحساب التعقيد الزمني لخوارزميّة يُحسب عدد العمليات من كل نوع.

لا توجد قاعدة ثابتة أو نظام متكامل يسمح بعدّ العمليات، إذ يتوقف هذا الأمر على القواعد المتّبعة في كتابة الخوارزميّة لكنّنا نورد الملاحظات التالية التي تفيد في حساب التعقيد الزمني لخوارزميّة:

آ - عند وجود العمليات في متتالية من التعليمات فإن عددها الكلّي هو مجموع عدد العمليات في كل تعليمة. مثلاً إذا كان $P(X)$ عدد العمليات الأساسية للتعليمة X فإن:

$$P(X1 ; X2) = P(X1) + P(X2)$$

ب- عند وجود تفريع شرطي *(if then else)* فإنه من الصعب تحديد أي فرع سيتم تنفيذه، لكن يمكن إيجاد حد أعلى لعدد العمليات.

$$P(\text{if } C \text{ then } A \text{ else } B) \leq P(C) + \max(P(A), P(B))$$

جـ- عند وجود حلقات فإن عدد العمليات هو: $\sum_i P(i)$

حيث i هو المتحوّل الذي يتحكّم بالحلقة، و$P(i)$ عدد العمليات الأساسية المتعلّقة بالمرور رقم i في الحلقة.

د- فيما يتعلّق بالبرامج الجزئية *(التوابع والإجرائيات)* التي لا تستخدم العوديّة، فيكون عدد العمليات الأساسية الموافقة لاستدعاء برنامج جزئي هو عدد العمليات الأساسية الموجودة في هذا البرنامج الجزئي.

هـ- لحساب عدد العمليات اللازمة لتنفيذ برنامج جزئي عوديّ، يجب إيجاد طريقة لحل معادلة تراجعية. في الحقيقة يمكن التعبير عن عدد العمليات في الاستدعاء العوديّ لإجرائية مع قيمة n وليكن $T(n)$ بدلالة $T(k)$ حيث $k<n$ فمثلاً في خوارزميّة إيجاد العاملي لعدد صحيح موجب، إذا اخترنا عملية ضرب عددين صحيحين كعملية أساسية فإن:

$$T(0) = 0$$

$T(n) = T(n-1) + 1$ من أجل $n \geq 1$

وحل هذه العلاقة التراجعية البسيطة هو $T(n) = n$

2 - 3- التعقيد الزمني الوسطي والتعقيد في أسوأ الأحوال

من الواضح أن زمن خوارزميّة ما يتعلّق بالمعطيات التي تعالجها. وهذه المعطيات تتغيّر وفق مَنحيين: تغيّر في الحجم وتغيّر في المحتوى. ففي خوارزميّة البحث التسلسلي عن عنصر ضمن قائمة، يمكن أن يتغيّر كلٌّ من عدد عناصر القائمة *(حجم القائمة)* ومحتواها.

سنرمز بـ Dn إلى لمعطيات المسألة ذات الحجم n.

وبـ $CostA(d)$ إلى التعقيد الزمني للخوارزميّة A من أجل المعطاة d.

نسمي التعقيد الزمني في أحسن الأحوال للخوارزميّة A المقدار:

$$Min_A(n) = Min \{ Cost_A(d) \; ; \; d \in D_n \}$$

ونسمي التعقيد الزمني في أسوأ الأحوال المقدار:

$$Max_A(n) = Max \{ Cost_A(d) \; ; \; d \in D_n \}$$

ونسمي التعقيد الزمني الوسطي المقدار:

$$Average_A(n) = \sum_{d \in D_n} P(d).Cost_A(d)$$

حيث $P(d)$ احتمال أن تكون معطاة الخوارزميّة هي d. وعندما تكون جميع المعطيات متساوية الاحتمال تصبح العلاقة السابقة:

$$Average_A(n) = \frac{1}{\|D_n\|} \sum_{d \in D_n} Cost_A(d)$$

حيث $\| D_n \|$ عدد المعطيات ذات الحجم n.

مثال

سنحاول تحديد عدد المقارنات اللازمة للبحث التسلسلي عن عنصر ضمن قائمة تحوي n عنصراً باستخدام الخوارزمية A المشروحة في الفصل الأول.

من الواضح أن:

$$MaxA(n) = n$$

$$MinA(n) = 1$$

لحساب $AverageA(n)$ يجب معرفة بعض الاحتمالات حول القائمة L والعنصر X:

ليكن q احتمال أن يكون العنصر X موجوداً ضمن القائمة.

نفترض أنّه إذا وجد X في السلسلة فإن المواضع كلها متساوية الاحتمال.

من أجل $1 \leq i \leq n$، سنرمز بـ Dn,i إلى مجموعة كل القوائم التي طولها n والتي يظهر فيها X أول مرّة في الموقع رقم i، وسنرمز بـ $Dn,0$ إلى مجموعة القوائم التي لا يظهر فيها X .

بحسب الفرضيات السابقة:

$$P(D_{n,i}) = \frac{q}{n}$$

من تحليل الخوارزميّة نستنتج أن:

$$Cost(Dn,0) = n$$

$$Cost(Dn,i) = I$$

ومن ثمَّ:

$$Average_A(n) = \frac{q}{n}\sum_{i=1}^{n} i + (1-q).n = \frac{q}{n}\frac{n(n+1)}{2} + (1-q).n = \frac{q(n+1)}{2} + (1-q).n$$

فإذا كنّا نعلم سلفاً أن X موجود ضمن L فإن:

$$Average_A(n) = \frac{(n+1)}{2}$$

وإذا كان احتمال وجود X ضمن L هو 0.5 فإن:

$$Average_A(n) = \frac{3(n+1)}{4}$$

2 - 4- مقارنة الخوارزميات

بعد تحديد تعقيد خوارزميّة كتابع لحجم المعطيات، يمكن دراسة سرعة تزايد هذا التابع عندما يزداد حجم المعطيات. تفيد هذه الدراسة في تحديد فعّالية الخوارزميّة من أجل معالجة معطيات كبيرة الحجم، إذ يمكن في بعض الحالات أن نجد فروقاً هائلة بين خوارزميّتين من حيث التعقيد الزمني.

في معظم الحالات نكتفي بتقريب بسيط لتابع التعقيد الزمني، لمعرفة فعّالية الخوارزميّة ولمقارنة خوارزميّتين. فمثلاً عندما تكون n كبيرة يكون من غير المهم أن نعرف، أتحتاج خوارزميّة معيّنة إلى n أم إلى $n+5$ عملية. ويمكن في معظم الأحوال إهمال الثوابت الضربية في تابع التعقيد. فمثلاً إذا كنّا نريد مقارنة الخوارزميّة A التي تعقيدها الزمني $M_A(n) = n^2$ مع الخوارزميّة B ذات التعقيد الزمني $M_B(n) = 4\,n$ فإن الخوارزميّة B أفضل من أجل جميع قيم n تقريباً (من أجل $n>4$).

تؤول التقريبات السابقة إلى إيجاد ما يسمّى مرتبة كبر تابع، وتجري مقارنة الخوارزميّات على أساس مرتبة الكبر لتوابع التعقيد الزمني.

تعريف

نقول عن التابع f إنه مهمل تقاربياً *(Asymptotic Neglected)* أمام التابع g ونرمز إلى ذلك بـ $f = o(g)$ (أو $f(n) = og(n)$) إذا كان:

$$f/g \rightarrow 0 \text{ when } n \rightarrow \infty$$

تعريف

نقول عن التابع f إنه مكافئ تقاربياً *(Asymptotic Equivalent)* للتابع g ونرمز إلى ذلك بـ $f \approx g$ (أو $f(n) \approx g(n)$) إذا كان:

$$f/g \rightarrow 1 \text{ when } n \rightarrow \infty$$

تعريف

نقول عن التابع f إنه مُسيطر عليه تقاربياً *(Asymptotic Dominated)* من قبل التابع g ونرمز إلى ذلك بـ $f = O(g)$ (أو $f(n) = Og(n)$) إذا كان:

$$\exists \text{ constant } c > 0 : |f(n)| \leq cg(n) \text{ when } n \rightarrow \infty$$

تعريف

نقول عن التابع f إنه من نفس مرتبة التابع g ونرمز إلى ذلك بـ $f = \Theta(g)$ (أو $f(n) = \Theta g(n)$) إذا كان:

$$f = O(g) \text{ and } g = O(f)$$

أمثلة

$$f(n) = n^2 + n = O(n^2) \;;$$

$$f(n) = 5n^3 = O(n^3) \;;$$

$$p(n) = a_m n^m + a_{m-1} n^{m-1} + \ldots + a_1 n + a_0,\ a_m \neq 0 \Rightarrow p(n) = O(n^m) \;; \; p(n) = o(n^{m+1}) \;;\; p(n) = \Theta(n^m)$$

ملاحظات

- قد تكون إشارات المساواة في التعاريف السابقة مربكة. لذلك منعاً للالتباس تجدر الإشارة إلى أن الرموز $o(g)$ و $O(g)$ و$\Theta(g)$ تعني عملياً:

$$o(g) = \{ f : f \text{ asymptotic neglected to } g \}$$

$$O(g) = \{ f : f \text{ asymptotic dominated by } g \}$$

$$\Theta(g) = \{ f : f = O(g) \text{ and } g = O(f) \}$$

لذلك المساواة $f = o(g)$ تعني عملياً $f \in o(g)$

- من الواضح أن: $f = o(g) \Rightarrow f = O(g)$ وأن $f \approx g$ إذا وفقط إذا

$f = g + o(g)$

- $f = \theta(g)$ إذا وفقط إذا وجِد ثابتان موجبان $c1$ و $c2$ بحيث عندما $n \to \infty$ يكون $c1g(n) \leq f(n) \leq c2g(n)$

نستطيع الآن بعد هذه التذكرة ببعض المفاهيم الرياضية وضع الأساس العملي لحساب تعقيد الخوارزميات:

لتكن A و B خوارزميتين تحلان نفس المسألة ولنفترض أننا استطعنا إيجاد تابعين f و g بحيث:

$CostA(n) = O(f(n))$; $CostB(n) = O(g(n))$

عندئذ نستطيع مقارنة الخوارزميتين A و B بمقارنة التابعين f و g.

لتكن A و B خوارزميتين تحلان نفس المسألة ولنفترض أننا استطعنا إيجاد تابعين f و g بحيث:

$CostA(n) = O(f(n))$; $CostB(n) = O(g(n))$

عندئذ نستطيع مقارنة الخوارزميتين A و B بمقارنة التابعين f و g.

هذه الطريقة تبسط تحليل تعقيد الخوارزميات وذلك بإهمال العوامل الثابتة التي يعود معظمها إلى خواص نموذج الحساب.

2 -5- أمثلة

5 - 1 - المثال الأول: حساب تعقيد خوارزمية الفرز بالفقاعات

ليكن لدينا الجدول $A[1..n]$ الذي يحوي n عنصراً، ونريد حساب تعقيد خوارزمية الفرز بالفقاعات *(Bubble Sort)*:

الإجرائية

```
procedure BubbleSort (n: integer; var A: array[1..n] of integer);
var
        i, j : integer;
begin
        for i:=n-1 downto 1 do
        for j:=1 to i do
        if A[j+1] < A[j] do
        begin
                {swap A[j] and A[j+1]}
                t:=A[j];
        A[j]:=A[j+1];
        A[j+1]:=t;
        end
end;
```

الحل

بُعد المسألة هو n. ويحتوي البرنامج على حلقتين (واحدة داخل الأخرى). هناك $n-1$ مروراً في الحلقة الخارجية. في المرور $n-i$ نجري i عملية مقارنة وi عملية تبديل مواقع ولذا:

طريقة أخرى

$$Cost(n) = \sum_{i=1}^{n} 2i = n(n-1) = \theta(n^2)$$

في المرور الأول نجري $n-1$ عملية مقارنة و$n-1$ عملية تبديل مواقع في أسوأ الأحوال. في المرور الثاني نكرر نفس العملية لكن على جدول $A[1..n-1]$ ومن ثم نستنتج العلاقة التراجعية:

$T1 = 0;$

$T(n) = T(n-1) + 2(n-1);$

هذا يؤدي إلى:

$$Cost(n) = \sum_{i=1}^{n} 2(n-i) = \sum_{j=1}^{n-1} 2j = n(n-1) = \theta(n^2)$$

5 - 2 - المثال الثاني

نريد حساب تعقيد خوارزمية إعادة تمثيل عدد A من الأساس D إلى الأساس العشري حيث $D \leq 10$:

$$A = a_n D^n + a_{n-1} D^{n-1} + \dots + a_1 D + a_0$$

الإجرائية (الطريقة المباشرة)

```
function Direct (n: integer; A: array[1..n] of integer): integer;
var
        i, j, C, P : integer;
begin
        C:=A[0];
        for i:=1 to n do
        if A[i] <> 0 then
        begin
        P:=A[i];
                for j:=1 to i do
                P:=P*D;
                C:=C+P
        end;
        Direct:=C;
end;
```

الحل

بُعد المسألة هو n درجة كثير الحدود. العمليات الأساسية هي عمليتي الضرب. في الحلقة الداخلية هناك i عملية ضرب وعملية جمع واحدة. وبالتالي عدد عمليات الجمع sum وعدد عمليات الضرب $prod$ يحسب كالأتي:

الطريقة الثانية (طريقة هورنر Horner)

$$sum = \sum_{1}^{n} 1 = n$$
$$prod = \sum_{i=1}^{n} i = \frac{n(n-1)}{2} = \theta(n^2)$$

نكتب A كما يلي:

$$A = ((\ldots((a_n D + a_{n-1}) \times D + a_{n-2}) \times D + \ldots) \times D + a_1) \times D + a_0$$

للحصول على C يكفي الحساب في الأساس العشري للمتتالية $(C)_{1 \le i \le n}$ حيث:

$$C_0 = a_n;$$

$$C_i = C_{i-1} \times D + a_{n-i}$$

وباعتبار أن: $C_n = C$ فإننا نجري n عملية ضرب فقط. ويكون التعقيد من رتبة $\theta(n)$ (لاحظ التحسين الذي أجريناه مقارنةً مع الطريقة المباشرة).

يمكن أن نقول إن الخوارزميّات الجيّدة لحل مسألة معيّنة هي التي يكون زمن تنفيذها:

- **ثابتاً مهما كان حجم المعطيات:** مثل خوارزميّات البحث في جداول التقطيع *(Hashing Tables)*.

- **لوغارتمياً:** كخوارزميّات البحث الثنائي.
- **خطّياً:** كخوارزميّة البحث التسلسلي التي عرضناها سابقاً.
- **من مرتبة $n.log_2(n)$:** كخوارزميّات الفرز الجيّدة.

يمكن القول بوجهٍ عام، إن الخوارزميّات التي يكون زمن تنفيذها من مرتبة n^k (والتي تسمى عادةً خوارزميات كثيرات الحدود) حيث: $K> 0$ ليست فعّالة إلاّ إذا كانت $K<2$، وعندما تكون $2<K<3$ فإننا نستطيع معالجة معطيات متوسّطة الحجم. وعندما يكون $K>3$ فإننا لا نستطيع أن نعالج إلا مسائل ذات حجم معطيات صغير.

أما الخوارزميّات ذات التعقيد الأسّي (2^n مثلاً) فلا تُستخدم إلاّ في حالة معطيات ذات حجم محدود ولهذا وصفناها بأنها غير فعّالة.

2 - 6 - تمارين

1- ضرب كثيري حدود

ليكن كثيرا الحدود $A(X)$ و $B(X)$:

$$A(X) = a_0 + a_1X + \dots + a_{n-1}X_{n-1}$$

$$B(X) = b_0 + b_1X + \dots + b_{n-1}X_{n-1}$$

آ- احسب التعقيد الزمني لحساب: $P(X) = A(X) \times B(X)$ بالطريقة المباشرة:

```
procedure DirectMult(n: integer; A,B: array[0..n-1] of real);
var
        i, j : integer;
begin
        for i:=0 to 2*n-2 do

        P[i]:=0;
        for i:=1 to n-1 do

        for j:=1 to n-1 do

        P[i+j] := P[i+j]+A[i]*B[j]
end;
```

ب- هل تستطيع إيجاد طريقة ثانية لحساب هذا الجداء بحيث تخفض من قيمة التعقيد الزمني.

2 - بفرض x عدداً كسرياً و n عدداً صحيحاً موجباً. يمكن حساب xn باستعمال أحد التابعين $power1$ و $power2$ أوجد التعقيد الزمني لهاتين الخوارزميتين من حيث عدد عمليات الضرب.

```
function power1 (x : real;  n : integer) : real;
var
        p : real;
begin
        p := 1;
        for  i := 1 to  n  do

        p := p * x;
        power1 := p
end;
```

```
function power2 (x : real;  n : integer) : real;
var
        factor, res : real;
begin
        res :=1;
        factor := x;
        while n > 0 do
        begin
        if (n mod 2) = 1 then
        res := res * factor ;
        factor:= factor * factor;
        n := n div 2
        end;
        power2 := res
end;
```

3- قدّر زمن تنفيذ الإجرائيات التالية:

```
procedure Try1;
var
        I, J, K, N, M, P, L : integer;
begin
        readln(N); M:=0; P:=0; L:=0;
        for I:=1 to N-1 do
        for J:=I+1 to N do
        for K:=1 to J do
        begin
        M:=M+1;
        P:=P*2;
        L:=L+3;
        end; {For}
        writeln(M, P, L)
end; {Try1}
```

```
procedure Try2;
var
        X, Y, I, J, N : integer;
begin
        readln(N);
        X:=0; Y:=0;
        for I:= 1 to Y do

        if odd(I) then

        begin

        for J:=1 to N do

        X:= X+1;

        for J:=1 to I do

        Y:= Y*2;

        end; {if}
        writeln(X, Y)
end; {Try2}
```

```
procedure Try3;
var
        X, Y, I, J, N : integer;
begin
        readln(N);
        X:=0; Y:=0;
        for I:= 1 to N do

        if (I mod 3 = 0) then

        for J:=I to N do

        X:= X+3

        else

        for J:=1 to I do

        Y:= Y*3;
        writeln(X, Y)
end; {Try3}
```

4 - ليكن لدينا التابع P التالي:

```
function P (x, n : integer) : integer;
Var term, result : integer;
Begin
        term  := x;   result := 1;
        While  n > 0 do

        Begin

        If ( (n mod 2) =1 ) then

        result := result * term ;

        term:= term * term ;

        n := n div 2

        End;
        P := result
End;
```

1- ما الذي يحسبه هذا التابع

2- ما هي درجة التعقيد الزمني لهذا التابع

5 - انظر إلى التابع *MySin* التالي، إنه يحسب جيب الزاوية x (بتقريب معين) باستخدام سلسلة تايلور:

$$\sin(x) = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \frac{x^9}{9!} - ..+..-..$$

يتوقف الحساب عندما تصبح القيمة المطلقة للحد *(term)* أقل من مقدار معين *epsilon*

```
1.    public static float MySin (float x)
2.    {
3.         flaot epsilon = 0.00000001;
4.         flaot term, result;
5.         int n, factn;
6.         result = 0.0;
7.         term = x;
8.         n = 1;
9.         factn = 1;
10.        for ( n=1 ; abs(term)> epsilon; n++ )
11.        {
12.        result  = result + term;
13.        factn = factn * (n+1) * (n+2);
14.        term = - term * x * x / factn;
15.        n= n+2;
16.        }
17.        return result;
18.   }
```

المطلوب:

1 - بفرض n عدد المرات التي يتم فيها المرور في الحلقة *for*. احسب عدد عمليات الضرب اللازمة لحساب جيب الزاوية x بهذه الطريقة بدلالة n

2 - أوجد درجة تعقيد هذا التابع بدلالة n

3 - إذا علمت أن $n!$ هو من مرتبة nn حاول حساب n بدلالة x و *epsilon*

4 - بين كيف يمكن تقليل عدد عمليات الضرب في السطرين 13 و 14

الفصل الثالث

الخوارزميات العودية

3 - 1- المقدمة

نقول عن شيء إنه ذو بنية عوديّة إذا كان مؤلفاً من مجموعة من المكونات بعضها مُعرّف تعريَف الشيء الأصليّ. نقول مثلاً إنّ الشجرة هي بنية عوديّة لأنها مؤلفة من مجموعة من الفروع كلٌّ منها يحوي فروعاً جديدة وبالتالي يكون له بنية الشجرة نفسها أو يحوي مجموعة من الأوراق.

من الأمثلة الشهيرة لاستخدام العوديّة في التعاريف نورد مايلي:

1- **تعريف الأعداد الطبيعية:** تُعرّف الأعداد الطبيعية بالشكل التالي:

آ- العدد صفر هو عدد طبيعي

ب- كل عدد يلي عدداً طبيعياً هو عدد طبيعي

2- **تعريف بنية الشجرة الثنائية:** تُعرّف الشجرة الثنائية بالشكل التالي:

آ- الشجرة O لاتحوي أي عقدة. ندعوها الشجرة الفارغة.

ب- إذا كانت A و B شجرتين ثنائيتين فإن $<X,A,B>$ هي شجرة ثنائية جذرها X وشجرتها الجزئية اليسارية A وشجرتها الجزئية اليمينية B.

3- تعريف العاملي لعدد صحيح غير سالب:

آ- $0! = 1$

ب- إذا كان $n>0$ فإنّ $n! = n.(n-1)!$

4- تعريف متتالية أعداد فيبوناتشي (Fibonacci):

تُعرّف هذه المتتالية بالشكل التالي:

آ- $Fib0 = 0 \;,\; Fib1 = 1$

ب- إذا كان $n>1$ فإنّ $Fibn = Fibn\text{-}1 + Fibn\text{-}2$

تكمن أهمية العوديّة في إمكانية تعريف مجموعة غير منتهية من الأشياء بواسطة مجموعة منتهية من التعليمات. بالطريقة نفسها يمكن تعريف عمليات حسابية غير منتهية بواسطة خوارزميّة عوديّة. تكون الخوارزميّات العوديّة أكثر ملاءمة عندما تكون المسألة المطلوب حلها أو بنية المعطيات الواجب معالجتها معرّفة بشكل عوديّ.

بشكل عام يمكن التعبير عن خوارزميّة عوديّة P بواسطة تركيب C لمجموعة عمليات أساسية Si (لا تحوي P) مع P نفسها:

$$P = C[Si, P]$$

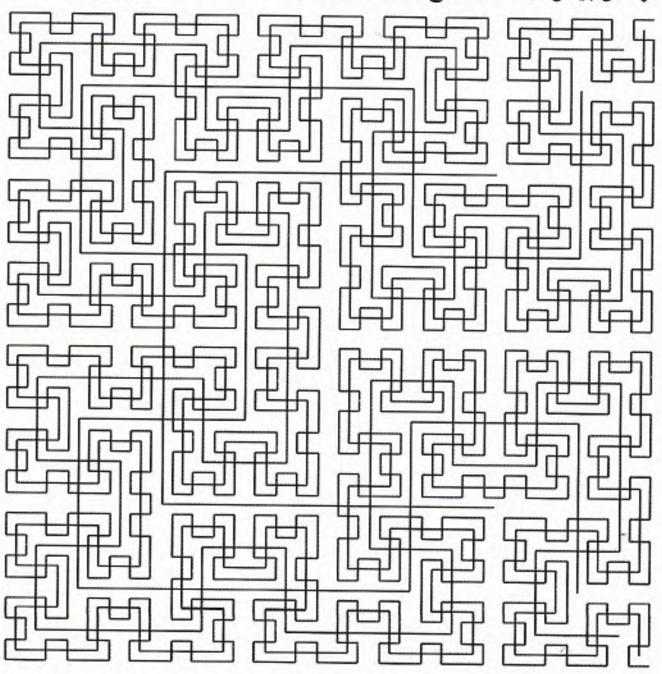
إنّ الأداة اللازمة والكافية للتعبير عن برنامج معيّن بشكل عوديّ هي الإجرائية *(Procedure)* لأنها تسمح بإعطاء اسم معين لمجموعة تعليمات ما يسمح باستدعاء هذه التعليمات بشكل عوديّ. يمكن التمييز بين نوعين من الإجرائيات العوديّة:

- **الإجرائيات ذات العودية المباشرة:** نقول عن إجرائية P إنها عوديّة بشكل مباشر إذا كانت تحوي استدعاءً صريحاً لنفسها.
- **الإجرائيات ذات العودية غير المباشرة:** نقول عن إجرائية P إنها عوديّة بشكل غير مباشر إذا كانت تستدعي إجرائية أخرى Q التي تستدعي P (بطريق مباشر أو غير مباشر).

3 - 2 - مثال لبرنامج عوديّ

إذا نظرنا بتمعن إلى الأشكال الزخرفية المبينة في الشكل *3-1* نرى أنها تتألف من خمسة خطوط منكسرة تتبع منهجاً في الرسم نستطيع برمجته بعد تحديده بدقة. لو نظرنا مرة أخرى إلى هذه الخطوط، وحاولنا عزل الثلاثة الأولى منها نجد أن لها شكلاً مبيناً بالشكل *3-2*، نرمز لهذه الأشكال بـ- *H1*، *H2*، *H3* على الترتيب.

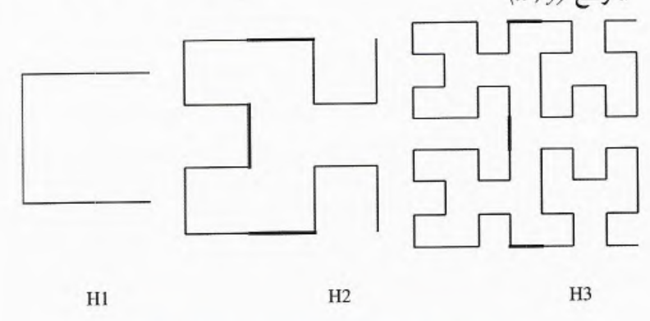
يُظهر الشكل *3-2* كيف نستطيع الحصول على المنحني *Hi+1* بتركيب أربعة منحنيات *Hi* أصغر حجماً بعد تدويرها باتجاهات معينة والوصل بينها بثلاث قطع مستقيمة رسمناها بخط عريض.



شكل *3-1*: منحنيات هلبرت من *H1* إلى *H5*

لنفرض أن الأدوات الأساسية المتوفرة للرسم هي:

- جملة إحداثيات *XOY*
- إجرائية *Setplot* تقوم بنقل قلم الرسم إلى الموقع *(x , y)*
- إجرائية *Plot* تقوم برسم خط مستقيم من موقع قلم الرسم إلى الموقع *(x , y)*



طريقة تركيب منحنيات هلبرت من *H1* إلى *H3*

بما أن كل منحني *Hi* يتألف من أربعة منحنيات *Hi-1* متصلة فإننا نستطيع التعبير عن إجرائية رسم *Hi* كتركيب لأربع إجرائيات تقوم كل منها برسم *Hi-1* بالاتجاه والقياس المناسبين. إذا رمزنا لهذه الإجرائيات الأربعة بالأحرف *A*، *B*، *C*، *D* وإذا رمزنا بأسهم لعمليات رسم الوصلات، فإننا نستطيع التعبير عن المخطط العوديّ لهذه الإجرائيات كما يلي:

A D ← A ↓ A → B

B C ↑ B → B ↓ A

C B → C ↑ C ← D

D A ↓ D ← D ↑ C

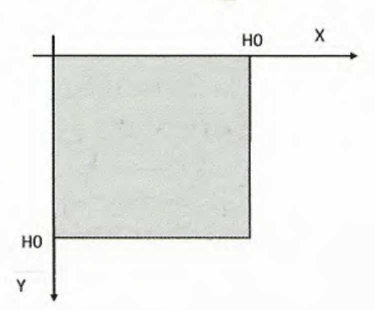
إذا كانت h طول القطعة المُستخدمة في رسم الخط المنكسر، فإن كتابة الإجرائية A ستأخذ التركيب التالي لـ D و B مع A:

```
procedure A (i: integer);
        begin
        if i > 0 then
        begin
        D(i - 1);     x := x - h;     plot;
        A(i - 1);     y := y - h;     plot;
        A(i - 1);     x := x + h;     plot;
        B(i - 1);
        end
end;
```

لتحديد نقطة بداية الرسم سنفرض أن منطقة الرسم عبارة عن مربع طول ضلعه $h0$، عندئذ سنلاحظ أن رسم الخط Hi سيبدأ من الموقع:

$$x = h_0 + \frac{h_0}{2^i}$$

$$y = h_0 - \frac{h_0}{2^i}$$



أما طول القطعة المُستخدمة في رسم الخط Hi فهو:

$$h = \frac{h0}{2^i}$$

بهذا ننهي تفصيل الخوارزميّة ويصبح برنامج رسم منحنيات هلبرت كالتالي:

```
program Hilbert;
const
   n  = 5;
   h0 = 512;
var
   i, h, x0, y0, x, y: integer;

procedure A (i: integer);
 begin
   if i > 0 then
     begin
       D(i - 1);        x := x - h;        plot;
       A(i - 1);        y := y + h;        plot;
       A(i - 1);        x := x + h;        plot;
       B(i - 1);
     end
 end;

 procedure B (i: integer);
 begin
   if i > 0 then
     begin
       C(i - 1);        y := y - h;        plot;
       B(i - 1);        x := x + h;        plot;
       B(i - 1);        y := y + h;        plot;
       A(i - 1);
     end
 end;
```

```
procedure C (i: integer);
begin
  if i > 0 then
    begin
      B(i - 1);     x := x + h;     plot;
      C(i - 1);     y := y - h;     plot;
      C(i - 1);     x := x - h;     plot;
      D(i - 1);
    end
end;

procedure D (i: integer);
begin
  if i > 0 then
    begin
      A(i - 1);     y := y + h;     plot;
      D(i - 1);     x := x - h;     plot;
      D(i - 1);     y := y - h;     plot;
      C(i - 1);
    end
end;

begin
  i := 0;
  h  := h0;
  x0 := h div 2;
  y0 := x0;
  repeat
     i := i + 1;
     h := h div 2;
     x0 := x0 + h div 2;
     y0 := y0 - h div 2;
     x := x0;
     y  := y0;
     Setplot;
     A(i)
  until i = n;
end.
```

3 - 3 - الخوارزميات التراجعية

الخوارزميات التراجعية *(Backtracking Algorithms)* هي خوارزميات عوديّة تعتمد طريقة "التجريب والخطأ" *(Try-and-Error)* في حل المسائل.

تتلخص هذه الطريقة ببناء الحل النهائي للمسألة عن طريق مجموعة من الخطوات. في كل خطوة نحدد الإمكانات المتاحة للخطوة التالية، ثم ندرس هذه الإمكانات بأن ننتقي أحدها، ونسجلها على أنها الخطوة التالية في الحل النهائي. ونتابع الخوارزميّة اعتماداً على هذه الخطوة. عندما يتأكد لنا أنّ اختيارنا لا يقود إلى الحل النهائي، أو يؤدي إلى طريق مسدود، نَعدِل عن هذه الخطوة. تشبه هذه العملية بناء سجل أخطاء يفيد في تجنب الوقوع في الخطأ مرتين.

يمكن اعتماد المخطط الخوارزمي التالي لحل المسائل المشابهة:

```
procedure try ;
begin
        initialize selection of candidates;
        repeat
        select next candidate;
        if acceptable then
        begin
        record it ;
        if solution incomplete then
        begin
        try next step ;
        if not successful then
        cancel recording
        end
        end
        until successful or no more candidates
end;
```

تختلف تفصيلات هذا المخطط باختلاف المسألة المطروحة. سنتداول هذا المخطط في المثال التالي من خلال مسألة معروفة هي مسألة جولة حصان الشطرنج، ونعرض في التمارين المحلولة حالات أخرى لاستخدامه.

لدينا رقعة شطرنج فيها $n \times n$ مربعاً. نضع حصاناً في موقع معين $<X0, Y0>$ حيث: $1 \leq X0 \leq n$ و $1 \leq Y0 \leq n$ وعلينا إيجاد طريقة لتغطية الرقعة (إذا كان ذلك ممكناً) أي أن نوجد متتالية من الحركات عددها $n^2 - 1$ بحيث يحدث المرور بكل مربع مرة واحدة فقط.

لحل المسألة سنهتم فقط بحركة الحصان التالية: سنفترض في كل مرحلة أننا قمنا بعدد من الحركات، ولدينا مجموعة من الإمكانات التي يجب أن نجربها. في كل مرحلة لدينا عدد من الحركات الممكنة. نجرّب إحدى هذه الحركات ونناقش أتؤدي إلى حل أم إلى طريق مسدود ؟

يمكن التعبير عن ذلك بالخوارزميّة العوديّة المبسطة التالية والمستوحاة من المخطط الخوارزمي التراجعي العام:

```
procedure try next move ;
begin
        initialize selection of moves;
        repeat

        select next candidate from list of next moves;

        if acceptable then

        begin

        record move;

        if board not full then

        begin

        try next move;

        if not successful then

        erase previous recording

        end

        end
        until (move was successful) or (no more candidates)
end;
```

لتفصيل الخوارزميّة سنحدّد بنى المعطيات المستخدمة ومعاملات الإجرائية. نمثل الرقعة بمصفوفة مربعة h نعرفها بالشكل:

```
const
n = 8;
type
index = 1..n;
var
h : array[index, index] of integer;
```

حيث نعتبر:

$h[X,Y] = 0$ عندما لا يحدث المرور في المربع (X, Y)

$h[X,Y] = i$ عندما يحدث المرور في المربع (X,Y) في الخطوة i

معاملات الخوارزمية

- الموقع الذي نريد تطبيق الخوارزميّة فيه: (X,Y)
- رقم الخطوة: i
- متحول خرج q يعطي نتيجة المناقشة: نجاح أو فشل

بناءً على ذلك يمكن تبسيط العبارة: "الرقعة ليست ممتلئة" (Board not full) بالشكل: $i < n^2$

إذا استعملنا متحولين موضعيين u, v لتمثيل أحد المواقع الممكنة (حسب طريقة حركة الحصان المعروفة على شكل L) فإنّ العبارة "مقبول" (Acceptable) يمكن التعبير عنها بالشكل:

$h[u, v] = 0$ and $1 \leq u \leq n$ and $1 \leq v \leq n$

أخيراً نُدخل متحولاً موضعياً آخر $q1$ لمراقبة الاستدعاء العوديّ للإجرائية، فتصبح الإجرائية كالتالي:

```
procedure try (i: integer; x, y: index; var q: boolean);
var
        u,v : integer; q1 : boolean;
begin
        initialize selection of moves;
        repeat
        let u,v be the coordinates of the next
        move defined by the rules of chess;
        if (1<=u<=n) and (1<=v<=n)
        and (h[u,v]=0) then
        begin
        h[u,v] := i;
        if I < sqr(n) then
        begin
        try(i+1,u,v,q1);
        if q1 then
        h[u,v] := 0
        end
        else
        q1 := true
        end
        until q1 or (no more candidates);
        q :=q1
end;
```



المرحلة الأخيرة من تبسيط الخوارزميّة هي تحديد الحركات الممكنة التي يمكن إجراؤها من موقع معين $<X, Y>$. نعلم من قواعد لعبة الشطرنج أن الحصان يستطيع في الحالة العامة الانتقال إلى ثمانية مواقع يبينها الشكل التالي.

		3		2			
	4				1		
			X				
	5				8		
		6		7			

يمكن الحصول على إحداثيات هذه المواقع الجديدة بإضافة فروق إلى إحداثيات موقع الحصان ونخزن هذه الفروق في جدولين a و b يُعرّفان بالشكل:

```
var
a, b : array[1..8] of integer;
```

كما نستطيع استخدام عداد k لترقيم إمكانات الانتقال التالي. عند استدعاء الإجرائية نحدد المعاملين $<X, Y>$ اللذين يمثلان الموقع الابتدائي للحصان، ونسند القيمة واحد للموقع $h[X, Y]$.

البرنامج التالي يعطي الحل الكامل لمسألة جولة الحصان على رقعة شطرنج، حيث يبدأ الحصان من الموقع $h[1, 1]$.

```
program knightstour (output);
const
        n = 5; nsq = 25;
type
        index = 1..n;
var
        i,j         : index;
        q           : boolean;
        s           : set of index;
        a,b         : array [1..8] of integer;
        h: array [index, index] of  integer;

procedure try (i: integer; x,y: index; var q: boolean);
        var

        u,v,k: integer; q1: boolean;
begin
        k := 0;
        repeat

        k := k+1; q1 := false;

        u := x + a[k]; v := y + b[k];

        if (u in s) and (v in s) then

        if  h[u,v] = 0 then

        begin
```

```
            h[u,v] :=i;
            if i < nsq then
            begin
            try (i+1,u,v,q1);
            if not q1 then
            h[u,v] := 0
            end
            else
            q1 := true
            end
            until q1 or (k=8);
            q := q1
end {try} ;

begin { program }
            s := [1,2,3,4,5];
            a[1] :=  2; b[1] :=  1;
            a[2] :=  1; b[2] :=  2;
            a[3] := -1; b[3] :=  2;
            a[4] := -2; b[4] :=  1;
            a[5] := -2; b[5] := -1;
            a[6] := -1; b[6] := -2;
            a[7] :=  1; b[7] := -2;
            a[8] :=  2; b[8] := -1;
            for i := 1 to n do
            for j := 1 to n do
```

```
        h[i,j] := 0;
        h[1,1] := 1;
        try(2,1,1,q);
        if q then
        for i :=1 to n do
        begin
        for j := 1 to n do write(h[i,j]:5);
        writeln
        end
        else
        writeln ('No Solution')
end. { Program }
```

3 - 4 - تمارين محلولة

3 - 4 - 1 - مسألة الوزراء الثمانية

الغرض من هذه المسألة هو توزيع ثمانية وزراء على رقعة شطرنج، بحيث لا يكون فيها أيّ واحد مهدداً للآخر. بحسب المخطط العام للخوارزميّات التراجعية فإنّ شكل الإجرائية هو:

```
procedure try(i : integer);
begin
        initialize selection of positions
        for i-th queen;
        repeat
        make next selection;
        if safe then
        begin
        setqueen;
        if i < 8 then
        begin
        try (i+1);
        if not successful then
        remove queen
        end
        end
        until  successful or no more positions
end { try };
```

نعلم من قواعد الشطرنج أنه يمكن للوزير أن يهدد جميع المربعات الموجودة في العمود أو السطر أو القطر نفسه، كما يبين ذلك الشكل *(3-4)*. لذا يجب وضع وزير واحد في كل عمود. وتؤدي عملية تحديد موقع الوزير رقم i إلى تحديد رقم السطر الذي سيوضع فيه ضمن العمود رقم i.

لحلّ هذه المسألة على الرقعة لابدّ من مناقشة عميقة لبنية المعطيات المستخدمة، بحيث تُحدَّد بنية تساعد في إجراء الاختبارات اللازمة عند تقرير وضع الوزير في مربع معين. في البداية يمكن تعريف رقعة الشطرنج بأنها مصفوفة مربعة، لكن هذا التمثيل سيؤدي إلى اختبارات صعبة لنعرف أيكون مربع ما آمناً *(غير مهدَّد)* أم لا.

إنّ ما يهمنا هو موقع كل وزير ضمن العمود الموافق، وأن نعرف أيحوي سطرٌ معين أو قطر معين وزيراً أم لا. لذا سنستخدم بنى المعطيات التالية:

```
x : array[1..8]    of integer;
a : array[1..8]    of boolean;
b : array[2..16]   of boolean;
c : array[-7..7]   of boolean;
```

حيث:

- $x[i]$ تحوي رقم السطر الذي يوضع فيه الوزير رقم i
- $a[j]$ تعني أن السطر j لا يحوي أيَّ وزير
- $c[j]$ تعني أن القطر المباشر رقم j لا يحوي أيَّ وزير
- $b[j]$ تعني أن القطر المتعامد رقم j لا يحوي أيَّ وزير

بعد تحديد بنى المعطيات المُستخدمة نستطيع تفصيل الخوارزميّة. فنلاحظ أن المواقع الممكنة للوزير رقم i هي كل مربعات العمود رقم i (عددها ثمانية). لاختبار هذه المواقع نستعمل عدّاداً j يكون في البداية صفراً ويزداد في كل مرة واحداً حتى يصل إلى القيمة 8. تكافئ عملية وضع الوزير رقم i في السطر j تنفيذ التعليمات التالية:

```
x[i]    := j;
a[j]    := false;
b[i+j]  := false;
c[i-j]  := false;
```

وتكافئ عملية رفع الوزير رقم i من السطر j تنفيذ التعليمات التالية:

```
a[j]    := true;
b[i+j]  := true;
c[i-j]  := true;
```

ويمكن التعبير عن القضية "أمان" *(Safe)* بالشكل:

```
a[j] and b[i +j] and c[i - j]
```

تنتهي بذلك عمليات تفصيل الخوارزميّة ونورد فيما يلي النص الكامل للبرنامج.

```
program EightQueens (output);
{ find one solution to eight queens problem }
var
        i : integer;
        q : boolean;
        a : array [1..8]  of boolean;
        b : array [2..16] of boolean;
        c : array [-7..7] of boolean;
        x : array [1..8]  of integer;

procedure try (i: integer; var q: boolean);
        var
        j : integer;
begin
        j := 0;
        repeat
        j := j+1; q := false;
        if a[j] and b[i+j] and c[i-j] then
        begin
        x[i]  := j;
        a[j]  := false;
        b[i+j] := false;
        c[i-j] := false;
        if i < 8 then
        begin
        try(i+1,q);
        if not q then
```

```
        begin
        a[j]   := true;
        b[i+j] := true;
        c[i-j] := true;
        end
        end
        else
        q := true
        end
        until q or (j=8)
end; { try }

begin { Program }
        for i := 1 to 8
        do a[i] := true;
        for i :=  2 to 16        do b[i] := true;
        for i := -7 to 7         do c[i] := true;
        try (1,q);
        if q then
        begin
        for i := 1 to 8          do write(x[i]:4);
        writeln
        end
        else
        writeln('No Solution')
end. { Program }
```

3 - 4 - 2- مسألة الخيار الأمثل

لإيجاد الحل الأمثل لمسألة معيّنة يجب إيجاد جميع الحلول الممكنة بحيث نحتفظ دوماً بحل واحد يُعتبر أمثليّاً بحسب معيار معين. إذا افترضنا أنّنا نستطيع مقارنة الحلول بواسطة تابع موجب $F(S)$ حيث (S أحد الحلول)، عندئذ يمكن الحصول على الحل الأمثل بتعديل المخطّط العام للخوارزميّات التكرارية التي توجد جميع الحلول بحيث نستبدل عملية إظهار الحل بالتعليمات التالية:

```
if f(Solution) > f(Optimal) then
Optimal := Solution
```

وبذلك نحتفظ في كل لحظة بأفضل حل أمكن الحصول عليه في المتحوّل *Optimal*.

سنختار مثالاً لمسألة الحل الأمثل مسألة هامّة تتلخّص بإيجاد الاختيار الأمثل من مجموعة عناصر معطاة. تجري عملية بناء الخيارات الممثّلة للحلول المقبولة بالتدريج، عن طريق مناقشة كل عنصر من عناصر المجموعة الأساسية. نناقش في حالة كل عنصر فائدة ضمّ هذا العنصر إلى الاختيار أو عدم فائدته، ثُمّ ننتقل إلى العنصر التالي عوديّاً.

عند فحص فائدة عنصر معيّن نستنتج أحد أمرين: إمّا أن نضم هذا العنصر إلى الخيار الآني الذي نحن بصدد بنائه، أو نستبعد هذا العنصر ونتابع بناء اختيارنا دونه. ويجب مناقشة هذين الاحتمالين كلٍ على حدة، وهذا ممّا يجعل خوارزميّة الحل تأخذ الشكل التالي:

```
procedure try (i : integer);
begin
          if inclusion is acceptable then

          begin

          include i-th object;

          if i < n   then

          try(i+1)

          else

          check optimality;

          eliminate i-th object

          end;
          if exclusion is acceptable then

          if i < n   then

          try (i+1)

          else

          check optimality
end;
```

تبيّن دراسة هذه الخوارزميّة أن هناك 2^n اختياراً ممكناً (عدد أجزاء مجموعة مؤلّفة من n عنصر) لذلك يجب أن نستفيد من معايير القبول في الحد كثيراً من هذا العدد. سنوضّح طريقة تقليص هذا العدد في ضوء مثال محدّد.

لتكن $A = \{a1, a2, . . ,an\}$ مجموعة من الأشياء لكل منها وزن W وقيمة V، ولنفترض أننا نريد بناء مجموعة جزئية من A بحيث يكون مجموع أوزان عناصرها أقل من حد معيّن، ومجموع أسعارها أعظمياً. تصادف هذه المسألة أي مسافر يستعد للسفر بالطائرة، ويقوم بإعداد حقيبته، وعليه اختيار مجموعة صغيرة من الأشياء لا يتجاوز وزنها العام الوزن المسموح به في المطار.

لتوضيح مبدأ العمل نأخذ حالة بسيطة، تحوي المجموعة فيها أربعة عناصر، يبيّن الجدول التالي وزن كل منها وقيمته.

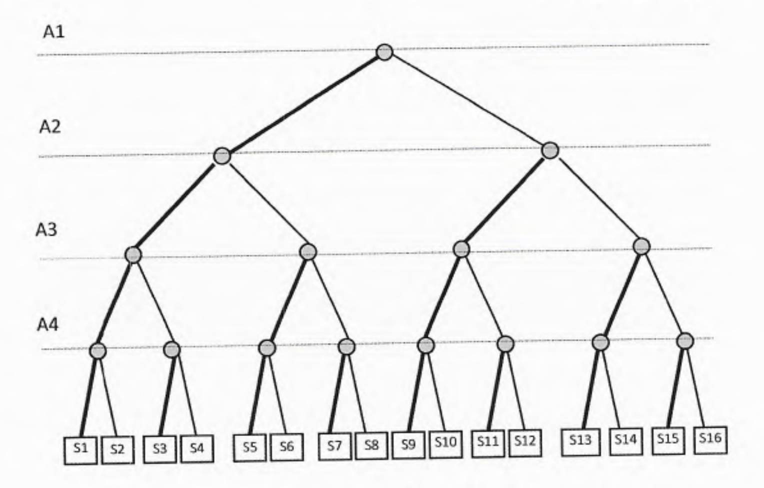
العنصر	$a1$	$a2$	$a3$	$a4$
الوزن (W)	10	11	15	17
القيمة (V)	18	20	25	17

ونريد اختيار مجموعة جزئية من هذه العناصر بحيث لا يزيد وزنها الكلّي عن 30 كغ وتكون قيمتها أعظمية. يمكن تمثيل تنفيذ خوارزميّة البحث عن أفضل اختيار بالشجرة المبيّنة بالشكل التالي، تمثّل كل عقدة داخلية في هذه الشجرة عملية مناقشة ضمّ أو استبعاد عنصر. *(رمزنا إلى ناتج ضمّ العنصر بخط عريض)*. وتمثّل الأوراق الخيارات الناتجة أو المجموعات الجزئية للمجموعة A *(عددها 16)*.

من دراسة هذه الحلول يتبيّن أنّه يجب استبعاد الحلول $S1, S2, S3, S5, S9$ لأنها لا تحقّق شرط الوزن. أمّا بقية الحلول فيجب حساب قيمة كل منها، ويستطيع القارئ التحقق من أن الحل $S10$ هو أفضل حلّ، ويعطي القيمة 45.

لنفكّر قليلاً في استنتاج طريقة لتقرير ضم عنصر إلى المجموعة الجزئية التي نحن بصدد بنائها أو عدم ضمّه. نرقّم العقد الداخلية للشجرة السابقة. تُشبه عملية بناء المجموعات الجزئية التجوّل ضمن

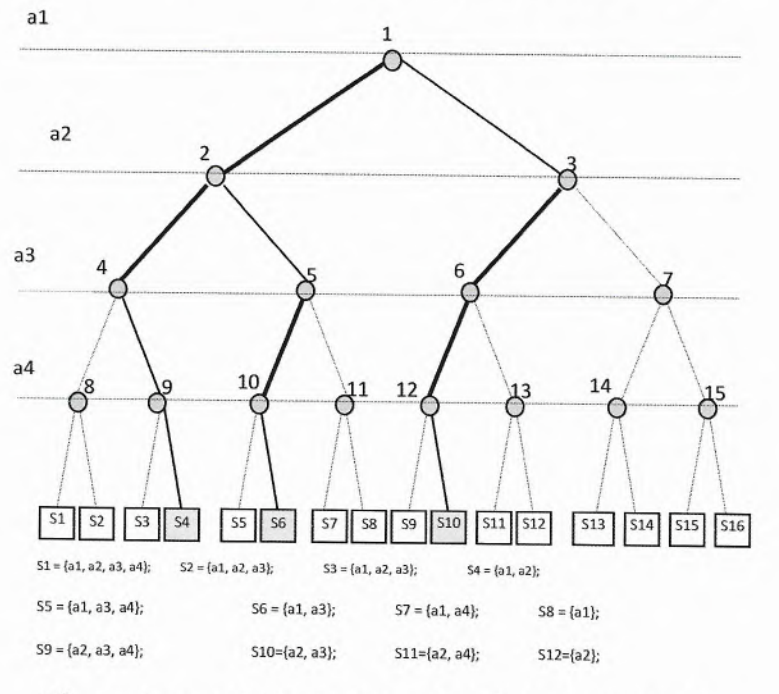
الشجرة السابقة بحيث نبدأ من العقدة رقم *1* ثمّ إلى العقدة رقم *2* ثمّ *4* ثمّ *8* وهكذا.



عندما نصل إلى العقدة *4* مثلاً، نكون قد أضفنا إلى المجموعة *S* التي نبنيها العنصرين *a1* و *a2*. ونناقش أنضم العنصر *a3* أم لا ؟ في هذه المرحلة نستطيع الاستغناء مباشرة عن الحلّين *S1* و *S2*، لأنّ ضمّ *a3* إلى المجموعة *S* سيجعل الوزن الكلّي لعناصر *S* يتجاوز الحد الأعلى. بمناقشة مماثلة نستنتج أن الحلول *S3, S5, S9* مرفوضة. عندما نصل إلى الحل *S4* (وهو أوّل حل مقبول من ناحية الوزن) نسجّل مجموع قيم عناصر وليكن *Maxv* (في هذه الحالة *Maxv = 38*).

عندما نصل إلى العقدة *5* تكون *S = {a1}* ونكون قد قرّرنا استبعاد العنصر *a2*. فنجري المحاكمة التالية: إن مجموع قيم العناصر التي ضُمّت حتى الآن هو *18*. وإذا استبعدنا *a3* فإن أفضل مجموع قيم سنحصل عليه وهو *18+17=35* أصغر من أفضل قيمة حصلنا عليها حتى الآن (قيمة الحل *S4*). ومن ثمَ فإن استبعاد العنصر *a3* مرفوض في

هذه المرحلة. بهذا نكون قد استبعدنا الحلول $S7$ و$S8$. بطريقة مشابهة تُستبعد الحلول من $S11$ إلى $S16$.



للتعبير عن طريقة المحاكمة السابقة على شكل خوارزميّة نعتبر أنّنا نقوم ببناء المجموعة S أي ستحوي حلاً مقبولاً. في كل خطوة سنناقش ضمّ عنصر واحد إلى S وسنفترض أن العناصر مرقّمة من 1 إلى n .

لنفترض tw الوزن الكلّي لعناصر S وأن av هو مجموع قيم عناصر S إضافةً إلى قيم العناصر التي لم نناقش بعد ضمّها إلى S. بافتراض i رقم العنصر الذي نعالجه حاليّاً.

في البداية $i = 1$:

$$Tw = 0; av = \sum_{i=1}^{n} a[i].v$$

عندما تصبح $i = n$ يكون:

$$Tw \le Limw; av = \sum_{a[i] \in S} a[i].v$$

عندئذ يمكن التعبير عن الشرط "الضم مقبول" بالقضية:

$tw + a[i].w <= limw$

ويمكن التعبير عن الشرط "الاستبعاد مقبول" بالقضية:

$av - a[i].v > maxv$

والذي يكافئ القول إن مجموع قيم العناصر الموجودة حالياً في S والتي يمكن ضمّها مستقبلاً يمكن أن يتجاوز أفضل قيمة وصلنا إليها حتى الآن. عندما تكون $i = n$ ويكون الاستبعاد مقبولاً، فإنه من المؤكد أن الحل الناتج أفضل من آخر حل أمكن الحصول عليه.

نجد كل هذه التفصيلات في نص الإجرائية *try* التي يتضمّنها البرنامج العام التالي. يبيّن الشكل التالي نتائج تنفيذ هذا البرنامج على مجموعة تحوي *12* عنصراً، حيث أعطينا الأوزان العظمى قيماً تقع بين *10* و *150* كيلوغراماً.

```
program Selection (input, output);
const
          n = 12;
type
          index = 1..n;
          objet = record

          v, w: integer
          end;
var
          i: index;
          a: array[index] of objet;
          limw, totv, maxv: integer;
          w1, w2, w3: integer;
          s, opts: set of index;
          z: array[boolean] of char;

procedure try (i: index; tw, av: integer);
var
          av1: integer;
begin
          if tw + a[i].w <= limw then

          begin

          s := s + [i];

          if i < n then

          try(i + 1, tw + a[i].w, av)

          else if av > maxv then

          begin

          maxv := av;

          opts := s

          end;

          av1 := av - a[i].v;
          if av1 > maxv then

          if i < n then

          try(i + 1, tw, av1)

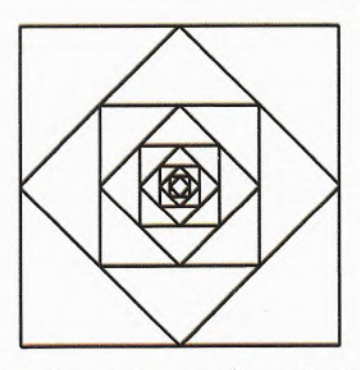
          else
```

```
            begin
            maxv := av1;
            opts := s
            end;
end;
begin { program }
            totv := 0;
            for i := 1 to n do
            with  a[i] do
            begin
            read(w,v);
            totv := totv + v
            end;
            read(w1, w2, w3);
            z[true]  := '*';
            z[false] := ' ';
            write('Weight ');
            for i := 1 to n do
            write(a[i].w : 4);
            writeln;
            write('Value');
            for i := 1 to n do
            write(a[i].v : 4);
            writeln;
            repeat
            limw := w1;
            maxv := 0;
            s := [];
            opts := [];
            try(1, 0, totv);
            write(limw : 5);
            for i := 1 to n do
            write(' ', z[i in opts]);
            writeln;
            w1 := w1 + w2
            until w1 > w3
end. { program }
```

الوزن	10	11	12	13	14	15	16	17	18	19	20	21
القيمة	15	18	15	17	21	20	19	21	23	24	24	25
10	*											
20										*		
30		*								*		
40		*			*	*						
50	*	*			*	*						
60	*	*	*	*	*							
70	*	*			*	*				*		
80	*	*	*		*	*			*			
90	*	*			*	*				*		*
100	*	*		*	*	*			*	*		
110	*	*	*	*	*	*		*	*			
120	*	*		*	*	*			*	*	*	
130	*	*	*	*	*	*		*	*	*		
140	*	*	*		*	*			*	*	*	*
150	*	*	*	*	*	*		*	*	*		*
160	*	*		*	*	*		*	*	*	*	*
170	*	*	*	*	*	*		*	*	*	*	*
180	*	*		*	*	*	*	*	*	*	*	*
190	*	*	*	*	*	*	*	*	*	*	*	*

3 - 5 - تمارين غير محلولة

1 - اكتب إجرائية عوديّة تقوم برسم الأشكال الزخرفية Pn الناجمة من تراكب عدد من الأشكال التي تشبه النموذج. يبين الشكل التالي المنحني $P5$.



يُشترط في إجرائية الرسم أن ترسم المنحني المطلوب دون رفع القلم ودون رسم قطعة مستقيمة أكثر من مرة واحدة.

2 - مسألة الزواج المستقر

نفترض وجود مجموعتين A و B لهما نفس العدد n من العناصر. المطلوب إيجاد مجموعة من n ثنائية $<a, b>$ بحيث:

$$a \in A, b \in B$$

وتحقق بعض الشروط. يمكن وضع العديد من الشروط على هذه الثنائيات؛ لكن الشرط الذي يهمنا في هذه المسألة هو "شرط الزواج المستقر".

لتكن A مجموعة من الرجال وB مجموعة من النساء. كل رجل وكل امرأة يستطيع تحديد أفضليات من أجل اختيار الشريك الذي يناسبه.

إذا تم اختيار الـ n زوجاً (أي ثنائية) وبقي على الأقل رجل وامرأة ويفضل كلاهما الآخر على الشريك الذي اختير بنتيجة الخوارزمية نقول عن الزواج إنه ليس مستقراً. أما إذا لم يوجد مثل هذه الحالة فنقول عن هذا الزواج إنه مستقرٌ.

لاحظ أن هذه المسألة تميز مجموعة هامة من المسائل مثل مسألة المفاضلة في الجامعة أو مسألة فرز الموظفين إلى المؤسسات.

نفترض أيضاً أن قوائم أفضليات كل شخص تكون جاهزة قبل تنفيذ الخوارزمية ولا يجوز تعديلها أثناء عمل الخوارزمية.

3 - إحدى شركات القطارات تُخدّم n محطة: $S1, S2, . ., Sn$ وترغب في تقديم خدمة معلومات لزبائنها. تتلخص هذه الخدمة بإتاحة طرفيات للمستثمرين في محطات القطارات. يستطيع المستثمرون بواسطتها الاستفسار عن كيفية الذهاب من أي محطة Sd إلى محطة أخرى Sa. إجابة برنامج هذه الخدمة يجب أن تكون قائمة بالقطارات التي تصل المحطة Sd بالمحطة Sa (ليس بالضرورة وجود قطار مباشر بين المحطتين. ونفترض أيضاً مراعاة زمن كاف للتبديل بين قطارين) بأسرع وقت ممكن.

4 - اكتب خوارزمية تراجعية لحل مسألة المتاهة وفق الفرضيات التالية:

- تمثل المتاحة بمصفوفة h أبعادها $n x n$ تحوي إحدى القيمتين صفر (ويرمز إلى عدم إمكانية المرور بهذا المربع) أو واحد (يمكن المرور بهذا المربع)
- نفترض أننا نريد الوصول من المربع $(1,1)$ إلى المربع (n,n)
- يمكن التحرك أفقياً أو شاقولياً فقط (لايمكن التحرك بشكل قطري)



الفصل الرابع

مدخل إلى قواعد المعطيات

4 - 1 - المقدمة

قاعدة المعطيات ***(Database):*** مجموعة من المعطيات المهيكلة غير المتكررة، المسجلة على وسط تخزين يسمح بالوصول إليها من قبل عدة برامج تطبيقية.

نظام إدارة قواعد المعطيات ***(DBMS: Data Base Management Systems):*** تجمع من المعطيات المرتبطة فيما بينها، ومجموعة من البرامج التي توفر الوصول إلى هذه المعطيات.

الهدف الأساسي لنظم إدارة قواعد المعطيات هو توفير محيط عمل ملائم وفعال يمكّن من تخزين المعلومات ضمن قاعدة المعطيات واسترجاعها لاحقاً. وقد صُممت هذه النظم لإدارة كميات ضخمة من المعلومات.

تتضمن إدارة المعلومات المهام الرئيسية التالية:

- تعريف بنى تخزين المعلومات.
- إيجاد التقنيات الملائمة للتعامل مع المعلومات المُخزنة.
- تقديم نظم أمان لحماية المعلومات المخزنة من الوصول غير المشروع.
- تجنب التضارب في المعلومات المخزنة نتيجة تشارك عدة مستثمرين في الوصول إلى المعلومات.

4 - 2 - الغرض من نظم إدارة قواعد المعطيات

تسعى نظم إدارة قواعد المعطيات لحل المشكلات الأساسية التي تواجه تطوير البرمجيات بالطرق التقليدية. أهم هذه المشكلات:

- **تكرار المعطيات وتضاربها**

يمكن أن تستغرق عمليات تعريف الملفات وكتابة البرامج التطبيقية مدة طويلة، وقد يعمل في ذلك مبرمجون مختلفون، وقد تظهر مشاكل عديدة: فالملفات وُلِّدت بأشكال مختلفة، ويمكن أن تكون البرامج التطبيقية قد كُتبت بلغات برمجة مختلفة أيضاً. وأكثر من ذلك يمكن أن تكون المعلومات نفسها مكررة في أكثر من ملف.

إن وجود مثل هذا التكرار يسبب هدراً في حجم التخزين، وكلفة عالية في الوصول إلى المعطيات، ويمكن أن يؤدي إلى معطيات متضاربة، وذلك أن وجود عدة نسخ من المعطيات في ملفات مختلفة لا تبقى متوافقة مدة طويلة (حدوث تعديل أو حذف في مكان دون آخر).

- **صعوبة الوصول إلى المعطيات**

لا يسمح محيط إدارة الملفات باسترجاع المعطيات المطلوبة بطريقة فعالة، ولابد من كتابة برامج عديدة لمعالجة الاستفسارات المختلفة. لذلك يصبح من الضروري تطوير نظام عام يفيد في استرجاع المعطيات استرجاعاً أفضل.

- **عزل المعطيات**

إذا كانت المعطيات موزعة في عدة ملفات ذات بنى مختلفة، يصبح من الصعب كتابة تطبيق جديد لاسترجاع المعطيات وفق أشكال معينة. إن سبب هذه المشكلة هو أن تعريف المعطيات يجري ضمن البرامج التي تدير هذه المعطيات. ومن ثمّ فإن أي تعديل في بنى التخزين يجب أن يواكبه تعديل كل البرامج التي تتعامل مع المعطيات.

● **تعارض في الوصول المتزامن**

تسمح النظم المعلوماتية الكبيرة لأكثر من مستخدم بالوصول إلى المعطيات لإجراء عمليات الإضافة والحذف والتعديل والاستفسار. وذلك بُغية الحصول على زمن استجابة أقصر وزيادة مردود هذه النظم. ولكن ذلك يزيد من أخطار التعديل المتزامن للمعطيات ويزيد تضارب المعطيات الناجّة عنه.

● **أمن المعطيات**

يُقصد بأمن المعطيات قدرة النظام على تحديد صلاحيات الوصول إلى المعطيات. فمثلاً في النظام المصرفي يحتاج المسؤول عن دفع رواتب موظفي المصرف إلى معرفة معلومات عن موظفي المصرف، ولا يحتاج إلى معرفة معلومات عن الحسابات والزبائن المتعاملين مع المصرف. لتحقيق مثل هذه الإمكانات تحتاج طرق البرمجة التقليدية إلى إضافة تطبيقات عديدة إلى النظام. وفي بعض الأحيان يحتاج تحقيق بعض شروط الأمن إلى جهد يتجاوز الجهد المبذول في تحقيق الوظائف الأساسية للنظام.

● **تكامل المعطيات**

قد تخضع المعطيات المخزنة في النظام لِشروط معينة. فمثلاً يمكن أن يشترط المصرف أن رصيد أي حساب يجب ألّا يقلَّ عن *1000* ل.س، وأن الرصيد الأعظم لحساب التوفير هو مليون ليرة سورية. وينبغي أخذ مثل هذه الشروط بعين الاعتبار في جميع البرامج التطبيقية التي يتضمنها النظام. وكما نرى فإنه من الصعب جداً في نظام إدارة الملفات إضافة شرط جديد إلى المعطيات، لأن ذلك يتطلب تعديل جميع البرامج المكتوبة سابقاً، والتي تستخدم هذه الملفات. كما أنه من الصعب إضافة شروط متعلقة بمعطيات مختلفة مخزنة في ملفات مختلفة.

4 - 3 - وظائف أنظمة إدارة قواعد المعطيات

لحل المشاكل المذكورة آنفاً، ولتوفير إمكانات إضافية، جرى تطوير نظم إدارة قواعد المعطيات كطريقة عامة، تشمل مجموعة من المفاهيم وتوفر برمجيات عامة تفيد في تحقيق الأهداف التالية:

4 - 3 - 1 - مركزية المعلومات

تهدف قواعد المعطيات إلى تجميع كافة المعطيات المتعلقة بمؤسسة ما ضمن نظام واحد، يقوم بإدارة هذه المعطيات إدارةً قياسية، ويوفر جميع حاجات التطبيقات من المعطيات. يوفر اعتماد نظام معلومات مركزي في أي مؤسسة مزايا عديدة أهمها إلغاء التكرار، وتوفير سهولة إدخال وتحديث المعلومات، ومركزية التحكم والمراقبة.

4 - 3 - 2 - استقلال المعطيات

الهدف الأساسي لنظم إدارة قواعد المعطيات هو توفير الوسائل الكفيلة بجعل المعطيات مستقلة عن طريقة التخزين وعن البرامج التي تقوم بالتعامل مع هذه المعطيات. يجري تحقيق هذه الغاية بجعل التعامل مع المعطيات بواسطة برامج تقوم بالوصول إلى هذه المعطيات من مستوى عال من التجريد. لا يظهر الطريقة الفعلية للتخزين، ولا يحتاج إلى معرفة كافة التفاصيل المتعلقة ببنية القاعدة ومحتوياتها الشاملة.

إن تحقيق هذا الهدف بواسطة البرامج التي يتضمنها نظام إدارة قواعد المعطيات يخفف الأعباء الملقاة على عاتق المبرمجين، والمتمثلة في ضرورة تعديل البرامج التطبيقية لدى كل تعديل في بنى تخزين المعطيات، سواء في بنية الملفات أو في طريقة تنظيم الملفات أو وسائط التخزين. تتيح نظم إدارة قواعد المعطيات إمكاناً للتعامل مع المعطيات بقطع النظر عن بنيتها الداخلية، وتمكن البرامج

التطبيقية من متابعة العمل على المعطيات، في حال حدوث تغير في بنى التسجيلات لا يتناقض مع البنى التي كانت تستخدمها، ولا تتأثر البرامج التطبيقية بتغير طرق الوصول. فإذا جرت إضافة فهرس *Index*، أو دمج ملفان في ملف واحد، فإن ذلك لا يستدعي تغيير البرامج التي تدير المعطيات.

4 - 3 - 3 - معالجة المعطيات بواسطة لغات غير اجرائية

معظم مستثمري نظم إدارة قواعد المعطيات هم مستثمرون عاديون ليس لديهم فكرة سابقة عن لغات البرمجة. لذلك يجب توفير لغة يستطيع المستثمر بواسطتها أن يسأل قاعدة المعطيات أو يعدّل تلك المعطيات دون تحديد خوارزمية الوصول إلى تلك المعطيات، بل فقط بأن يصف المعطيات التي يريد المستثمر التعامل معها. يسمى هذا النوع من اللغات لغات غير إجرائية.

تعتبر هذه النقطة من أهم الأهداف التي يجب أن يحققها نظام إدارة قواعد المعطيات، فوجود لغة غير إجرائية عالية المستوى يسمح لأي مستثمر كان بأن يستفيد من إمكانات النظام بفعالية و سهولة.

4 - 3 - 4 - التسهيلات الخاصة بإدارة المعطيات

توفر نظم إدارة قواعد المعطيات الوسائل اللازمة للتعبير عن المعطيات (طريقة تعريفها وتخزينها) والوصول إليها وعرضها. تسمى هذه الوسائل أدوات إدارة قواعد المعطيات. وللحصول على إدارة فعالة وجيدة للمعطيات، يجري عادة حصر بعض هذه الأدوات بشخص واحد يدعى مدير النظام أو بعدة أشخاص يملكون امتيازات خاصة.

4 - 3 - 5 - الوصول إلى المعطيات بفعالية

تسعى أنظمة إدارة قواعد المعطيات لِزيادة عدد الإجراءات التي تنفذ في ثانية واحدة. وذلك بزيادة عدد المستخدمين الذين يستطيعون

الوصول إلى المعطيات بآن واحد. وإنقاص زمن الاستجابة (الزمن اللازم للحصول على جواب طلب ما). لتحقيق ذلك تتضمن هذه الأنظمة خوارزميات وطرقاً خاصة لتقسيم المصادر (الوحدة المركزية. وحدات الدخل / الخرج) بين المستثمرين تقسيماً عادلاً.

4 - 3 - 6 - التحكم في تكرار المعطيات

إن وجود إدارة مركزية للمعطيات تمكن من حل مشكلة تكرار المعطيات. وذلك بإعطاء الانطباع بأن كل مستخدم من مستخدمي قاعدة المعطيات يتعامل مع نسخة مستقلة من قاعدة المعطيات. وتوفير الأدوات التي تنسق بين العمليات التي يجريها المستخدمون على النسخة الوحيدة من المعطيات.

4 - 3 - 7 - تكامل المعطيات

يسمح نظام إدارة قواعد المعطيات بتحقيق أنواع عديدة من شروط التكامل نعرضها من خلال أمثلة:

- **تكامل وحدات المعطيات.**

مثال: لا يمكن فتح حساب مصرفي لزبون دون معرفة عنوانه

- **التكامل المرجعي.**

مثال: لا يمكن إجراء عمليات مصرفية على حساب قبل فتح الحساب

- **وشروط التكامل المعرَّفة من قبل المستخدم:**

مثال: الرصيد أكبر من 250 ل.س.

لتحقيق ذلك يوفر نظام إدارة قواعد المعطيات الإمكانات اللازمة لتعريف هذه الشروط من جهة. ولكشف وإيقاف جميع العمليات التي قد تؤدي إلى الإخلال بهذه الشروط من جهة أخرى.

4 - 3 - 8 - تقسيم المعطيات

يُقصد بتقسيم المعطيات السماح بتقسيم معطيات قاعدة ما بين عدة تطبيقات، بحيث يستطيع كل منها الوصول إلى المعطيات دون أن ينتظر تطبيقاً آخر.

4 - 3 - 9 - أمن المعطيات

توفر نظم إدارة قواعد المعطيات إمكان حماية بعض المعطيات الخاصة، بحيث أن مجموعة محددة هي فقط التي تستطيع الوصول إلى تلك المعطيات. فمثلاً لا يستطيع مدير قسم معين أن يطلع على رواتب كل العاملين في الشركة، بل على رواتب الموظفين العاملين في قسمه فقط.

4 - 4 - تصميم قواعد المعطيات

من وجهة النظر البرمجية، تتألف قاعدة المعطيات من تجمع من الملفات المترابطة، ومجموعة من البرامج التي تسمح بالوصول إلى المعطيات المخزنة فيها واسترجاعها وتعديلها. وتهدف نظم قواعد المعطيات إلى تقديم إمكان التعامل مع جزء من المعطيات واستخدامها بطريقة فعالة تجعلها بمتناول عدد كبير من المستخدمين.

تقود هذه الاعتبارات إلى تصميم بنى معطيات معقدة لتمثيل المعطيات، مع ضرورة إخفاء هذا التعقيد للسماح لأكبر عدد من المستخدمين، الذين لا يملكون خبرة واسعة في البرمجة، بالوصول إلى المعطيات.

لتحقيق ذلك، ولما كان مستخدمو قاعدة المعطيات ليسوا بالضرورة خبراء في استخدام الحواسيب والبرمجة، فإنه يجب إخفاء التعقيد الموجود في تلك البنى بتحقيق وجود عدة مستويات للتصميم تتوافق مع مستوى التفصيل الذي يمكن لكل فئة من المستخدمين التعامل معه.

لتحقيق هذه الأهداف جرى تحديد ثلاثة مستويات من التجريد تسمى مخططات *(Schema)* لتوصيف أي قاعدة معطيات. يجري في كل مستوى توصيف القاعدة ببعض التفصيل الإضافي، عن المستوى الأعلى. كما يقدم نظام إدارة قواعد المعطيات الوسائل الكفيلة بإيجاد الترابط بين هذه المستويات المختلفة. تهدف هذه النماذج إلى تبسيط تعامل المستخدمين مع المعطيات. ونبين فيما يلي شرحاً مبسطاً لهذه المستويات:

- **المخطط المفاهيمي:** يعتبر المخطط المفاهيمي تجريداً للواقع يعكس عناصر المعلومات التي ستقوم قاعدة المعطيات بإدارتها. يجري من خلال المخطط المفاهيمي توصيف المحتوى المعلوماتي للقاعدة دون التعرض لأساليب النمذجة اللاحقة أو للاستفسارات التي سيجريها المستخدمون. يسمح هذا الفصل بترجمة المخطط المفاهيمي إلى أنواع مختلفة من المخططات المنطقية.

  يجري التعبير عن المخطط المفاهيمي بأشكال عديدة، وتعتبر مخططات الكيانات والارتباطات *(ERD: Entity-Relationship Diagrams)* أحد أهم الطرق المتبعة في إنشاء مخطط المفاهيمي. يسمح هذا النموذج بتوصيف قاعدة المعطيات بشكل مخططات بيانية تتضمن الكيانات الداخلة في بنية النظام والارتباطات بينها.

- **المستوى المنطقي:** وصف لمحتوى قاعدة المعطيات بلغة قياسية تمكن مطوري التطبيقات ومستخدمي القاعدة من التعامل مع المعطيات على مستوى عال من التجريد، يجنبهم الخوض في التفاصيل المتعلقة ببنية الملفات وطرق الوصول إليها.يعتبر المخطط المنطقي نواة قاعدة المعطيات، وتمثيلاً معيارياً لمعطيات المؤسسة يعكس طبيعة المعطيات وخصائصها وارتباطاتها. في هذا المستوى من التجريد يجري تحديد مايلي:

- أنماط المعطيات البسيطة والمركبة المستخدمة في المؤسسة.
- ارتباطات هذه الأنماط بعضها ببعض، بما يعكس واقع عمل المؤسسة.
- قواعد تكامل المعطيات: الخصائص التي يجب أن تحققها المعطيات المخزنة في القاعدة.

يوفر كل نظام إدارة قواعد معطيات على الأقل نموذجاً للمعطيات، يسمح للمستخدمين بالتعامل مع المعطيات بأسلوب أقرب ما يكون إلى الواقع الذي أتت منه هذه المعطيات. يتضمن أي نموذج مفهومين أساسيين:

- طريقة تعريف المعطيات.
- العمليات التي يمكن تطبيقها على المعطيات.

في النموذج العلاقاتي، مثلاً، يمكن تعريف المعطيات بأنها مجموعة من العلاقات، وكل علاقة هي جدول يحوي عدداً من الأسطر والأعمدة . يحوي كل عمود قيماً تنتمي إلى مجال معين.

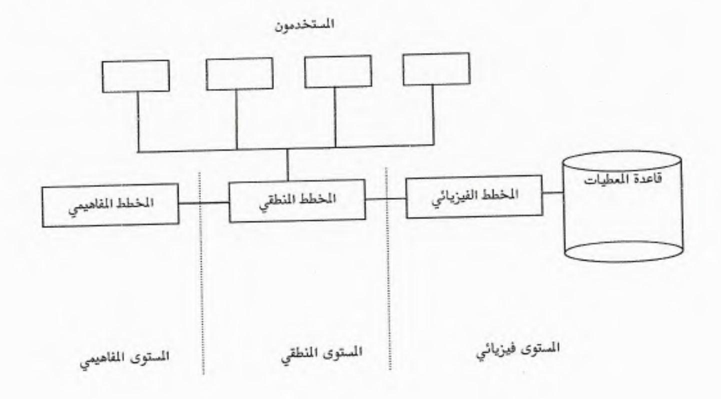
توفر هذه النماذج من المعطيات إمكان النظر إلى المعطيات من مستوى أعلى دون الخوض في طريقة التخزين الفيزيائي أو طرق الفهرسة، وتترك هذه المهام لِنظام إدارة قواعد المعطيات الذي يقوم بإيجاد ما يقابل هذه البنى في المستوى الفيزيائي. في نظم إدارة قواعد المعطيات، يجري التعبير عن بنية قاعدة المعطيات بواسطة لغة عالية المستوى تسمى لغة تعريف المعطيات *(DDL)*. تسمح هذه اللغة إضافةً إلى ماسبق بتعريف شروط تكامل المعطيات.

- **المستوى الفيزيائي (الداخلي) *Physical level***: وهو المستوى الأدنى في تجريد المعطيات، ويصف الطريقة الفعلية لتخزين المعطيات. يتعلق المستوى الداخلي ببنية التخزين التي ستحوي المعطيات فعلياً، ويسمح بوصف المعطيات حسب الطريقة المتبعة في التخزين:

  - توصيف ملفات قاعدة المعطيات *(أنماط الحقول وأطوالها، الحقول المركبة، ... الخ)*.
  - طرق التعامل مع وسط التخزين *(Segments, Blocks, Buffers, ... إلخ)*.
  - طرق الوصول إلى التسجيلات *(الفهرسة، ربط التسجيلات، ... إلخ)*.

غالباً لايحتاج القائمون على إدارة قواعد المعطيات إلى التدخل على هذا المستوى، ويتركون هذه المهمة لنظام إدارة قواعد المعطيات الذي يقوم بترجمة النموذج المنطقي إلى نموذج فيزيائي مكافئ.

في معظم الأحيان يجري تعريف مجموعات جزئية من المعطيات، تتضمن كل منها الجزء الذي يهم مستثمراً معيناً أو فئة من المستثمرين.



4 - 5 - لغات قواعد المعطيات

4 - 5 - 1 - لغة تعريف المعطيات

يوفر كل نظام إدارة قواعد معطيات على الأقل لغة واحدة تتيح لمستخدميه تعريف بنية قاعدة المعطيات، وشروط تكامل المعطيات وصلاحيات الوصول إلى المعطيات، وغيرها من التعاريف التي لابد منها لدى إنشاء القاعدة. وتوفر هذه اللغات طيفاً واسعاً من التعليمات التي تتيح للمستخدم والمبرمج إجراء عمليات الإضافة والحذف والتعديل والاستفسار.

وتوفر أنظمة إدارة قواعد المعطيات أدوات خاصة بالتطوير تمكن من تعريف واجهات التعامل مع قاعدة المعطيات (استمارات إدخال، لوحات تحكم، واجهات للاستفسار، تقارير، ... إلخ).

5-1- لغة تعريف المعطيات: *(DDL: Data Definition Language)*

يحدَّد مخطط قواعد المعطيات بمجموعة من التعاريف التي يُعبر عنها بلغة خاصة تسمى لغة تعريف المعطيات. إن نتيجة ترجمة تعليمات هذه اللغة هي مجموعة من الجداول المخزنة في ملفات خاصة تسمى قاموس المعطيات *(Data dictionary)*.

قاموس المعطيات هو ملف يحوي معطيات سامية *(meta-data)* أي "معطيات عن المعطيات" ويجري استدعاء هذا الملف قبل قراءة أو تعديل المعطيات الحقيقية في نظم قواعد المعطيات.

تتحدد بنى التخزين وطرق الوصول المستخدمة من قبل نظم قواعد المعطيات بمجموعة تعاريف في نوع خاص من لغة تعريف المعطيات، تسمى لغة تعريف وتخزين المعطيات. إن ترجمة هذه التعاريف تعطي مجموعة تعليمات تحدد التفاصيل التنفيذية لمخططات قواعد المعطيات التي لا تظهر للمستثمرين العاديين.

4-5-2 - لغة التعامل مع المعطيات *(DML: Data Manipulation Language)*

هي لغة تسمح للمستثمرين بالوصول والتعامل مع المعطيات المنظمة بنموذج معطيات معين. توفر لغة التعامل مع المعطيات الوظائف التالية:

- استخلاص المعطيات المخزنة في قواعد المعطيات
- إضافة معلومات جديدة إلى قاعدة المعطيات
- حذف معلومات من قاعدة المعطيات

يوجد نوعان رئيسيان من لغات التعامل مع المعطيات:

- **لغات إجرائية:** وتتطلب من المستثمر تحديد المعطيات التي يحتاج إليها وطريقة الحصول عليها.
- **لغات غير إجرائية:** وتتطلب من المستثمر تحديد المعطيات التي يحتاج إليها دون تحديد كيفية الحصول عليها.

اللغات غير الإجرائية أسهل تعلماً واستخداماً من اللغات الإجرائية. ولكن لأن المستثمر لايحدد كيفية الحصول على المعطيات، فيمكن أن تولّد اللغة طريقة للوصول ليست فعالة مثل التي يقوم بوضعها المستثمر بلغة التعامل الإجرائية. هذه الصعوبات يمكن أن نتجاوزها باستخدامنا لتقنيات اختزال متعددة.

4 - 6 - تمارين

1 - اذكر أربعة فروق أساسية بين نظم إدارة الملفات ونظم إدارة قواعد المعطيات

2 - اشرح الفرق بين ارتباط المعطيات المنطقي والفيزيائي

الفصل الخامس

المخطط المفاهيمي لقاعدة المعطيات

نموذج كيانات-ارتباطات

5 - 1 - المقدمة

يعتمد نموذج المعطيات كيان-ارتباط على تمثيل العالم الحقيقي بمجموعة من الأغراض تُسمى كيانات، وتعريف الارتباطات فيما بينها. طُوّر هذا النموذج لتسهيل تصميم قواعد المعطيات، فهو يسمح بتحديد مخطط المؤسسة الذي يمثل البنية المنطقية لقاعدة المعطيات.

5 - 2 - تعاريف أساسية

الكيان: هو غرض مميز عن غيره من الأغراض التي سيجري تخزينها في قاعدة المعطيات.

صفوف الكيانات: يجري تجميع الكيانات المتشابهة في مجموعات تسمى صفوف الكيانات. مثل: صف الأشخاص الذين لهم حساب في مصرف، صف الحسابات المصرفية، صف الموظفين، صف السيارات، صف القروض، .....

طريقة تمثيل الكيان: يُمثل الكيان بمجموعة من الواصفات. ولكل واصف مجموعة من القيم الممكنة. تُسمى مجموعة القيم الممكنة لواصفة بمجال الواصف *Attribute Domain*. من ثم يتصف الكيان بأزواج من الشكل *(واصف، قيمة الواصف)*.

يشبه مفهوم صف الكيانات مفهوم نمط المعطيات في لغات البرمجة. حيث يجري تعريف قالب عام يمثل مجموعة من الأغراض، يجري بواسطتها تعريف متحولات لكل منها قيمة معينة. ولكنها تشترك في البنية. إذ يوافق مفهوم المتحول في لغات البرمجة مفهوم الكيان في نموذج الكيانات والارتباطات.

الارتباط: هو علاقة تربط مجموعة من الكيانات بعضها ببعض. فمثلاً يمكن أن نجد ارتباطاً بين الشخص محمد والحساب المصرفي ذي الرقم *133*. يشير هذا الارتباط إلى أن محمداً هو زبون للمصرف وله حساب مصرفي رقمه *133*.

صف الارتباطات: هو مجموعة من الارتباطات من نوع واحد. يمكننا التعبير رياضياً عن صف الارتباطات كما يلي:

إذا كانت $E1, E2, E3, ..., En$ مجموعة من صفوف الكيانات فيعرف صف الارتباطات R كمجموعة جزئية من الجداء الديكارتي

$$\{ (e1, e2, e3, ..., en) \mid e1 \in E1, e2 \in E2, e3 \in E3, ..., en \in En \}$$

حيث $(e1, e2, e3, ..., en)$ هي علاقة ارتباط.

يمكن أن تتصف علاقة الارتباط بمجموعة من الواصفات. **مثال:** ربط الواصف «تاريخ» بعلاقة الارتباط زبون-حساب. يحدد هذا الواصف تاريخ الحالة التي أخذ فيها حساب الزبون المصرفي.

درجة الارتباط:

لتكن R علاقة ارتباط ثنائية بين صفي الكيانات A, B. تأخذ درجة الارتباط إحدى الحالات التالية:

واحد-واحد: كل كيان من صف الكيانات A يرتبط على الأكثر بكيان واحد من صف الكيانات B، وبالعكس كل كيان من صف الكيانات B يرتبط على الأكثر بكيان واحد من صف الكيانات A.

واحد-كثير: من الممكن لكيان من A أن يرتبط بأي عدد من الكيانات في B، وكل كيان من B يرتبط على الأكثر بكيان واحد من A.

كثير-واحد: كل كيان من A يرتبط على الأكثر بكيان من B ويمكن لكيان من B أن يرتبط بعدد من الكيانات من A.

كثير-كثير: يمكن لكيان من A أن يرتبط بعدد من الكيانات من B، وبالعكس يمكن لكيان من B أن يرتبط بعدد من الكيانات من A.

إن تحديد درجة الارتباط يعتمد على ملاحظة الواقع الذي تجري نمذجته بواسطة مجموعات الارتباطات. فإذا نظرنا إلى الارتباط بين صف الكيانات "زبون" وصف الكيانات "حساب" فإن هذا الارتباط يبدو للوهلة الأولى أنه من نوع واحد-واحد ولكن قد يتبين من مناقشة العاملين في المصرف أنه يُسمح لزبون معين أن يمتلك أكثر من حساب مصرفي. في هذه الحالة يتحول الارتباط زبون-حساب إلى النوع واحد-كثير، وفي مرحلة تالية يمكن أن تستنتج أن المصرف يسمح بفتح حسابات مشتركة (لأفراد العائلة الواحدة مثلاً) وعندها يتحول الارتباط إلى نوع كثير-كثير.

المفاتيح

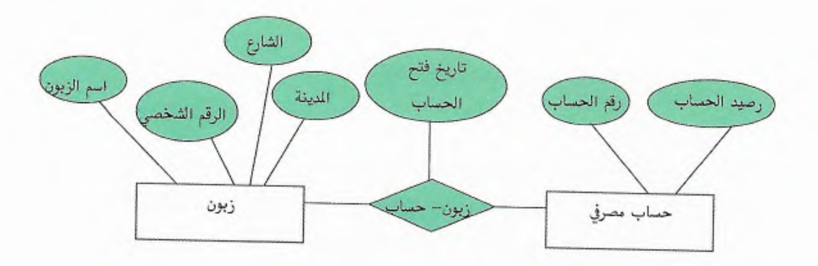
- **المفتاح الأعلى أو المفتاح الرئيسي *Superkey*:** مجموعة تضم واصفاً أو أكثر تسمح مجتمعة بتمييز كيان واحد من مجموعة من الكيانات المنتمية إلى صف واحد.
- **المفتاح المرشّح *Candidate key*:** وهو مفتاح رئيسي مؤلف من مجموعة من الواصفات ولا توجد مجموعة جزئية من هذه المجموعة تُكوِّن مفتاحاً رئيسياً.
- **المفتاح الأولي *Primary key*:** هو مفتاح مرشح اختاره مصمم قاعدة المعطيات كطريقة أساسية لتمييز الكيانات عن بعضها البعض والمنتمية إلى صف كيانات واحد.

صفوف الكيانات الضعيفة: نقول عن صف كيانات إنّه صف كيانات ضعيف إذا كان لا يحوي مفتاحاً أولياً.

5 - 3 - مخطط كيان-ارتباط *E-R Diagram*

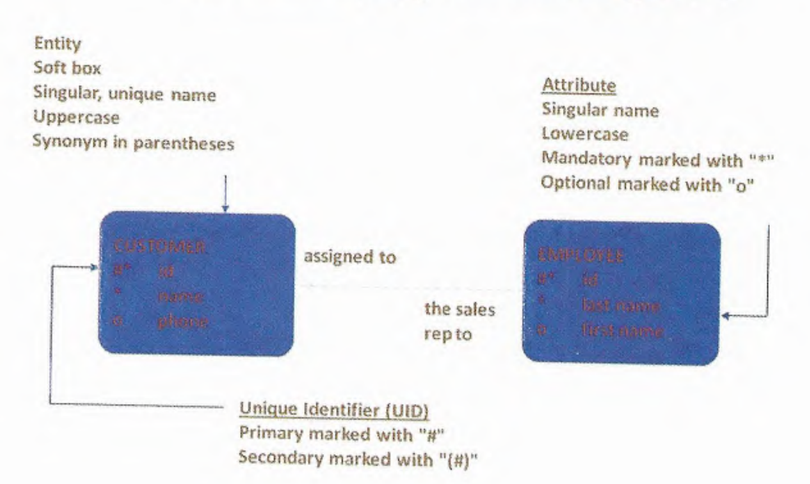
يمكن التعبير عن بنية قاعدة المعطيات بيانياً باستخدام مخطط كيان-ارتباط. يتألف هذا المخطط من المكونات التالية:

- **مستطيلات:** تُمثل صفوف الكيانات.
- **مستطيل مضاعف:** لتمثيل صف الكيانات الضعيف.
- **قطوع:** تُمثل الواصفات.
- **معينات:** تمثل صفوف الارتباطات.
- **خطوط:** تربط بين صفوف الكيانات بالواصفات، وتربط صفوف الكيانات بصفوف الارتباطات.
- خط تحت الواصفات المميزة لصف الكيانات.
- خط متقطع تحت الواصفات المميزة لصف الكيانات الضعيف.
- معيّن مضاعف لتمثيل علاقة ارتباط صف كيانات ضعيف بصف كيانات قوي.



مثال لمخطط كيانات -ارتباطات

5 - 4 - طريقة *Case Method* في رسم مخططات *ERD*



5 - 5 - تمارين

1- اشرح الفرق بين المفاهيم التالية: مفتاح أولي، مفتاح رئيسي، مفتاح مرشح

2- تتعامل مديرية التسجيل والامتحانات في كلية المعلوماتية مع معطيات حول الصفوف، والدوام، والمدرسين، وأوقات وأمكنة إعطاء الدروس. كما تحتفظ المديرية بالعلامة التي يحصل عليها الطالب في كل مادة وتقدير الطالب في كل سنة دراسية. المطلوب إعطاء المخطط كيان-ارتباط لهذه القاعدة.

3- لدى شركة تأمين مجموعة من الزبائن، يمتلك كل منهم سيارة أو أكثر. وتهتم شركة التأمين بحوادث السير التي تصيب كل سيارة من السيارات التي جرى التأمين عليها. المطلوب إعطاء المخطط كيان-ارتباط لهذه القاعدة.

4- لدى مشفى الأسد الجامعي عدد من المرضى، ويعمل فيه عدد من الأطباء. يتألف المشفى من عدد من الأقسام التخصصية، ويجري قبول المرضى كلٌّ في القسم المتخصص في حالة هذا المريض. وهناك أيضاً أقسام مشتركة تقدم الخدمات لكافة الأقسام مثل مخبر التحاليل الطبية، وقسم التصوير الشعاعي، والصيدلية. يخضع المريض خلال إقامته في المشفى لعدد من الفحوصات، وقد تجرى له عملية جراحية أو أكثر. لكل مريض من المرضى المقيمين في قسم معين طبيب مسؤول عن متابعته. ويكون هذا الطبيب واحداً من الأطباء العاملين في القسم.

المطلوب:

1. إعطاء المخطط كيان-ارتباط لهذه القاعدة

2. ماذا يحصل لدى انتقال المريض من قسم إلى آخر، وكيف يمكن تمثيل ذلك في قاعدة المعطيات ؟

3. كيف يمكن استرجاع السجل الطبي للمريض إذا راجع المشفى بعد مدة من خروجه من المشفى ؟

5- لدينا قاعدة معطيات خاصة ببرنامج الامتحان الأخير للجامعة. يمكن نمذجة هذه القاعدة كصف كيانات وحيد *exam* مع الواصفات *course_name,section_number, room_number, time*. كما يمكن بالمقابل أن نُعرف مجموعة من صفوف كيانات مرتبطة بروابط للاستعاضة عن بعض الواصفات في صف الكيانات *exam* بالشكل:

course مع الواصفات *name, departement, c_number*.

Section مع الواصفات *s_number, enrollment* ومرتبطة كصف كيانات ضعيف مع *course*.

Room مع الواصفات *r_number, capacity, building*

والمطلوب إعطاء مخطط *E-R* الذي يبين استخدام صفوف الكيانات الثلاثة المذكورة أعلاه.

6- نريد إنشاء مخطط *ERD* لقاعدة معطيات خاصة بشركة تأمين على السيارات. فيما يلي شرح لطريقة عمل شركة التأمين:

- لدى الشركة فروع منتشرة في كافة المحافظات، ويستطيع الزبون (الذي يريد إبرام عقد تأمين على سيارة) مراجعة أي فرع من هذه الفروع لتوقيع العقد.
- يمكن توقيع عقود التأمين مع أفراد أو مع شركات (مثل شركات نقل البضائع أو شركات نقل الركاب، أو حتى شركات تريد التأمين على سياراتها السياحية)

- بعد توقيع العقد يدفع الزبون في كل سنة بدل تأمين السنوي، وهو عبارة عن مبلغ يجري حسابه بناءً على نوع السيارة ونوع التأمين وفق ماهو مبين لاحقاً.

أ- توقيع العقود

تقدم الشركة عدة أنواع من عقود التأمين:

1. عقود التأمين على السيارات الخاصة.

هناك نوعان من عقود التأمين على السيارات الخاصة:

- تأمين إلزامي (أو تأمين الحد الأدنى، ويسمى أيضاً التأمين ضد الغير): تلتزم الشركة بموجبه بدفع الأضرار التي تسببها السيارة المؤمن عليها للغير دون أن تدفع قيمة الأضرار التي تصيب السيارة المؤمن عليها أو سائقها
- تأمين شامل على كل الأضرار التي تصيب السيارة أو سائقها أو تسببه السيارة للغير

يتعلق بدل التأمين السنوي بمايلي: استطاعة السيارة (سعة المحرك مقاسة بـ *CC*)، سنة الصنع، نوع التأمين (إلزامي أو شامل).

2. عقود التأمين على السيارات الشاحنة.

هناك نوع واحد من عقود التأمين على السيارات الشاحنة، ويتعلق بدل التأمين السنوي بما يلي: الوزن الفارغ للسيارة، الحمولة (طن)، سنة الصنع

3. عقود التأمين على سيارات نقل الركاب.

هناك نوع واحد من عقود التأمين على سيارات نقل الركاب، ويتعلق بدل التأمين السنوي بما يلي: مبلغ التعويض الأعظمي

الذي ستدفعه شركة التأمين لدى إصابة أو وفاة أحد الركاب، عدد الركاب، سنة الصنع.

ب- معالجة الحوادث

تهتم شركة التأمين بالحوادث التي التي تصيب كل سيارة من السيارات التي جرى التأمين عليها، ولدى وقوع حادث يجري تسجيل المعلومات التالية:

- نوع الحادث: اصطدام، تدهور، حريق، سرقة، ...
- مكان وقوع الحادث
- التاريخ
- نسبة مسؤولية سائق السيارة في الحادث
- رقم الضبط

يجري تقدير الأضرار الناجمة عن الحادث (الأضرار المادية وجسدية) وحساب المبالغ التي يجب أن تدفعها شركة التأمين سواءً للمؤمن عليهم (السيارة وسائقها وركابها) أو للغير (إذا تسببت السيارة بأضرار للغير).

المطلوب:

1. تحديد الكيانات وواصفات كل كيان والمفتاح
2. تحديد الارتباطات بين الكيانات
3. رسم مخطط *ERD*



الفصل السادس

النموذج العلاقاتي

6 - 1 - تعاريف

تتألف قاعدة المعطيات من مجموعة من الجداول لكل منها اسم وحيد مميز. يتألف كل جدول بدوره من مجموعة من الأعمدة وعدد من الأسطر. يمثل كل سطر من الجدول علاقة تربط مجموعة من القيم. لما كان الجدول يتألف من مجموعة من هذه الارتباطات، فهناك تشابه شديد بين مفهوم الجدول ومفهوم العلاقة الرياضية، ومن هنا أخذ النموذج العلاقاتي اسمه.

وفق مصطلحات النموذج العلاقاتي تسمى عناوين الأعمدة واصفات *attributes*. لكل واصف مجموعة من القيم تسمى مجال *Domain* الواصف. فمثلاً مجال الواصف "اسم الفرع" هو مجموعة كل أسماء الفروع. ولنرمز إلى هذه المجموعة بـ*D1*. ولنرمز بـ *D2* إلى مجال الواصف *account_number* وبـ *D3* إلى مجال الواصف *balance*.

يعرف الرياضيون العلاقة بأنها مجموعة جزئية من الجداء الديكارتي لمجموعة من المجالات.

6 - 2 - مخطط قاعدة المعطيات

يجري التمييز بين مخطط قاعدة المعطيات *database schema* أو التصميم المنطقي لقاعدة المعطيات وحالة قاعدة المعطيات *database instance* التي هي صورة لمعطيات قاعدة المعطيات في لحظة محددة.

يشابه مفهوم العلاقة مفهوم المتحولات في لغات البرمجة. على حين يشابه مفهوم مخطط العلاقة مفهوم تعريف الأنماط في لغات البرمجة.

6 - 3 - لغات الاستعلام

لغة الاستعلام هي اللغة التي يطلب بواسطتها المستخدم معلومات من قاعدة المعطيات. وهي عادة من مستوى أعلى من لغات البرمجة القياسية.

يمكن تصنيف هذه اللغات في نوعين: لغات إجرائية ولغات غير إجرائية.

- **اللغات الإجرائية:** يقوم فيها المستثمر بتحديد مجموعة من العمليات التي يجريها النظام على قاعدة المعطيات للوصول إلى المعطيات المرغوبة.
- **اللغات غير الإجرائية:** يصف فيها المستثمر المعلومات المرغوبة دون إعطاء الإجراء المحدد للحصول عليها.

تقدم معظم نظم قواعد المعطيات التجارية كلتا اللغتين.

6 - 4 - الجبر العلاقاتي

الجبر العلاقاتي هو لغة استعلام إجرائية. يتألف من مجموعة من العمليات التي تأخذ علاقة أو اثنتين كدخل، وتعطي علاقة جديدة كخرج. العمليات الأساسية في الجبر العلاقاتي هي:

- الاختيار *Select*
- الإسقاط *Project*
- الاجتماع *Union*

- الفرق *Difference*
- الجداء الديكارتي *Cartesian product*
- إعادة التسمية *Rename*

إضافةً إلى العمليات المعرفة ابتداءً من العمليات الأساسية:

- التقاطع *Intersection*
- الدمج الطبيعي *Natural Join*
- القسمة *Division*
- النَّسَب *Assignment*

سوف نعرض فيما يلي العمليات الأساسية والإضافية في الجبر العلاقاتي موضحين طريقة عملها بأمثلة على قاعدة المعطيات المتعلقة بالمصارف والمعرفة بالمخطط العلاقاتي التالي:

Loan_schema (loan_number, amount, branch_name)

Customer_schema(customer_name, customer_street, customer_city)

Borrower_schema(customer_name, loan_number)

Employee_schema(employee_name, phone_number)

Loan_officer_schema(banker_name, customer_name, loan_number)

Depositor_schema (customer_name, account_number)

Account_schema(account_number, balance)

Branch_schema(branch_name, branch_city, assets)

6 - 4 - 1- العمليات الأساسية

تسمى العمليات (اختيار، إسقاط، إعادة التسمية) عمليات أحادية لأنها تجرى على علاقة واحدة. العمليات الثلاث الباقية هي ثنائية لأنها تُجرى على زوج من العلاقات.

عملية الاختيار

تقوم باختيار مجموعة من الحدوديات التي تحقق شرطاً معيناً من علاقة. سنرمز إلى العملية كما يلي:

حيث *condition* هو شرط يجب أن تحققه الحدوديات المختارة.

$$\sigma_{condition}(relation)$$

مثال: لنأخذ مخطط العلاقة التالية

Loan_schema (branch_name, loan_number, amount)

لاختيار مجموعة الحدوديات القروض للفرع *branch_name="perryridge"* نكتب العبارة التالية:

$$\sigma_{branch\text{-}name="perryridge"}(loan)$$

يمكن أن يحوي الشرط المطبق في عملية الاختيار عمليات مقارنة : =، <>، <، >، =<، => ومعاملات منطقية *and, or, not* ويمكن أن تطبق هذه المعاملات بين قيم الواصفات المكونة للعلاقة.

مثال: لاختيار القروض التي منحها الفرع *branch_name="perryridge"* والتي لا تقل عن *1200* نكتب:

$$\sigma_{branch\text{-}name="perryridge" \wedge amount \geq 1200}(loan)$$

لاختيار مجموعة الزبائن الذين يحملون نفس اسم المسؤول عن القرض الذي اقترضوه من علاقة *loan_officer* ذات المخطط التالي

Loan_officer = (customer_name, banker_name, loan_number)

نكتب:

$$\sigma_{\text{customer-name,=banker-name}}(\text{loan - officer })$$

عملية الإسقاط *projection*

وهي عملية وحيدة المعامل تسمح بانتقاء بعض الواصفات من العلاقة. نرمز إلى هذه العملية بالشكل:

مثال: لنفترض أننا نريد الحصول على أرقام القروض ومبالغها دون أن نهتم بالأسماء الأفرع.

$$\prod_{selected \quad attributes}(relation)$$

$$\prod_{loan-number,amount}(loan)$$

يكتب الاستعلام السابق بالشكل:

وبفرض أن علاقة القروض *loan* ممثلة بالجدول التالي:

loan_number	*branch_name*	*amount*

تكون العلاقة الناتجة من عملية الإسقاط من الشكل:

loan_number	*amount*

تركيب العمليات العلاقاتية

إن نتيجة العمليات العلاقاتية هي علاقة. هذا ما يسمح لهذه العمليات أن تجتمع لتؤلف عبارات الجبر العلاقاتي.

مثال: لإيجاد أسماء الزبائن الذين يعيشون في مدينة *"Harrison"*.

نكتب:

$$\prod_{customer-namet}\left(\sigma_{customer-city="Harrison"}(customer)\right)$$

ونقصد بها تركيب عمليتين: اختيار الحدوديات التي تحقق الشرط (مدينة = *"Harrison"*) وإسقاط العلاقة الناتجة من العملية السابقة على العمود*customer_name*.

عملية الاجتماع *Union Operation*

نرمز إلى العملية بـ U وتُجرى هذه العملية بين علاقتين متجانستين فنقول إنه يمكننا القيام بالعملية $r\ U\ s$ إذا تحقق الشرطان التاليان:

- عدد الواصفات في العلاقتين r و s هو نفسه.
- مجال الواصفة رقم i في r هو نفسه مجال الواصفة رقم i في s.

ويمكن بالطبع أن تكون العلاقتان r , s ناجتين عن تعبير في جبر علاقاتي.

مثال: للحصول على جميع الزبائن الذين يتعاملون مع المصرف (الذين لديهم حساب أو اقترضوا قرضاً أو للحصول على كلا الفريقين).

نحتاج إلى إيجاد مجموعة الزبائن الذين لديهم حساب في المصرف، وهي معلومات موجودة في علاقة *depositor*، وإلى إيجاد مجموعة الزبائن الذين اقترضوا من المصرف. وهي معلومات موجودة في علاقة

borrower ثمّ إلى إجراء عملية الاجتماع بين المجموعتين.

ومن ثم يكون التعبير الناتج هو :

$$\prod_{customer-name}(depositor) \cup \prod_{custome-name}(borrower)$$

عملية الفرق *Difference Operation*

نرمز إلى هذه العملية بـ "−" وتسمح بإيجاد الحدوديات التي تنتمي إلى علاقة الحد الأول ولا تنتمي إلى علاقة الحد الثاني.

لإيجاد مجموعة الزبائن الذين لديهم حساب مصرفي ولم يقترضوا من المصرف نكتب:

$$\prod_{customer-name}(depositor) - \prod_{custome-name}(borrower)$$

وكما ذكرنا في عملية الاجتماع لكي تجري عملية الفرق بين علاقتين يجب أن يتحقق الشرطان المذكوران في الفقرة السابقة.

عملية الجداء الديكارتي *Cartesian Product*

نرمز إلى هذه العملية بـ X وتسمح بتجميع معلومات من علاقتين ونكتب $r1 \ X \ r2$ وبوجهٍ عام نقول:

إذا كان لدينا العلاقتان $r1(R1), r2\ (R2)$ فإن $r1 \ X \ r2$ هي علاقة R مخططها العلاقاتي هو تلاصق $R1$ و $R2$ وتحوي جميع الحدوديات t التي تحقق الشرط التالي:

يوجد $t1$ من $r1$ و $t2$ من $r2$ بحيث

$$t[R1] = t1[R1] \ \ \hat{}\Lambda \ \ t[R2] = t2[R2]$$

مثال:

- ليكن لدينا $R1$ و $R2$ بحيث $R= R1 X R2$

R

A	B	C	D
a1	b1	c1	d1
a1	b1	c2	d2
a2	b2	c1	d1
a2	b2	c2	d2

R1

A	B
a1	b1
a2	b2

R2

C	D
c1	d1
c2	d2

- إذا أردنا الحصول على جميع الزبائن الذين اقترضوا من المصرف الفرع *"perryridge"*.

للحصول على هذه المعلومات نحتاج إلى المعلومات الموجودة في كلتا العلاقتين *loan* و *borrower* وتعطي عملية اختيار الحدوديات، المتعلقة بالفرع المطلوب من جداء العلاقتين، معلومات عن الزبائن والقروض المأخوذة من الفرع المطلوب، ولكن ليست المعلومات المطلوبة، وللحصول على المعلومات المطلوبة نكتب:

$$\prod_{customer-name}\left(\sigma_{borrower-loan-number=loan-number}\left(\sigma_{branch-name="perryridge"}(borrower \times loan)\right)\right)$$

إعادة التسمية *Rename*

من المفيد إعطاء أسماء للعلاقات الناتجة عن تعبير جبر علاقاتي. نرمز إلى العملية بالرمز:

$$\rho_x(E)$$

التي تعني أن نتيجة التعبير E توضع في العلاقة x.

لنبين استخدام هذه العملية بالمثال التالي: لإيجاد الرصيد الأعلى في المصرف، نستخدم الاستراتيجية التالية:

- إيجاد مجموعة الأرصدة غير العظمى للحسابات في المصرف.
- طرح المجموعة الناتجة من مجموعة الأرصدة في المصرف فنحصل على الرصيد الأعظم المطلوب.

مجموعة الأرصدة الموجودة في المصرف هي:

$$\Pi_{balance}(account)$$

مجموعة الأرصدة غير العظمى هي:

$$\Pi_{account-balance}\left(\sigma_{account-balance<d.balance}\left(accountX\ \rho_{d}(account)\right)\right)$$

والنتيجة هي:

$$\Pi_{balance}(account)-\Pi_{account-balance}\left(\sigma_{account-balance<d.balance}\left(accountX\ \rho_{d}(account)\right)\right)$$

6 - 4 - 2 - العمليات الإضافية

عملية التقاطع

تُجرى هذه العملية بين علاقتين ويجب أن تحقق هاتان العلاقتان الشروط المذكورة في عملية الاجتماع. ونتيجة عملية التقاطع هي علاقة تحوي مجموعة الحدوديات الموجودة ضمن العلاقتين. التعبير الموافق لهذه العملية هو:

$$R\cap S=R-(R-S)$$

عملية الدمج الطبيعي *Natural join operation*

غالباً ما نرغب في تبسيط بعض الاستعلامات التي تحتاج إلى جداء ديكارتي، ومعظم عمليات الاستعلام التي تحوي جداءاً ديكارتياً تحوي عملية اختيار من نتيجة الجداء. فعملية الدمج هي عملية ثنائية تسمح بتركيب عملية الاختيار والجداء الديكارتي بعملية واحدة.

لنأخذ كتعريف لهذه العملية العلاقتين $r(R)$ و $s(S)$ ونقول إن دمج العلاقتين هي علاقة مخططها هو اجتماع مخططي العلاقتين ومعرفة بالشكل:

$$r \rhd\lhd s = \prod_{R \cup S} \left(\sigma_{rA!=sA1 \wedge rA2=sA2 \wedge ... rAn=sAn} (r \times s) \right)$$

حيث

$$R \cap S = \{A1, A2, ..., An\}$$

مثال:

لإيجاد أسماء جميع الأفرع التي لزبائنها حساب في المصرف وتعيش في مدينة *"Harrison"* نكتب.

$$\prod_{branch-name} \left(\sigma_{customer-city="Harrison"} (customer \rhd\lhd account \rhd\lhd depositor) \right)$$

حالات خاصة

- إذا كان $R \cap S = \Phi \Rightarrow r \rhd\lhd s = r \times s$

وكانت هناك قضية θ على الواصفات في مخطط العلاقة الناتجة عن الدمج نكتب:

$$r \rhd_{\Theta} \lhd s = \sigma_{\Theta} (r \times s)$$

عملية القسمة

هذه العملية مناسبة للاستعلامات التي تحوي كلمة «لأجل كل». فلإيجاد مجموعة الزبائن الذين لهم حسابات مصرفية في جميع الأفرع الموجودة في مدينة "brooklyn" نقوم بما يلي :

نستخرج مجموعة الفروع الموجودة في مدينة "brooklyn" بكتابة التعبير التالي:

$$r1 = \prod_{branch\text{-}name} \left(\sigma_{branch\text{-}city="brrolyn"} (branch) \right)$$

ثمّ نستخرج مجموعة الزبائن والفروع الذين لديهم حسابات فيها بكتابة التعبير:

$$r2 = \prod_{customer\text{-}name, branch\text{-}name} (depositor \rhd\lhd account)$$

ونحصل على النتيجة المطلوبة بكتابة:

$$r2 \div r1$$

عملية الإسناد

تعمل هذه العملية بكيفية مشابهة لعملية الإسناد في لغات البرمجة. وتعتبر طريقة مناسبة للتعبير عن استعلامات معقدة.

يرمز إلى العملية بـ " $\leftarrow$ "

6 - 5 - تمارين

لتكن قاعدة المعطيات العلاقاتية المعرفة بالمخططات العلاقاتية التالية:

employee (employee_name, street, city)

works(employee_name, company_name,salary)

company(company_name, city)

manages(employee_name, manager_name)

والمطلوب إعطاء التعبير الموافق بلغة الجبر العلاقاتي لكلٍّ مما يلي:

1. أسماء جميع الموظفين العاملين في المصرف التجاري.
2. أسماء ومدن إقامة جميع الموظفين العاملين في المصرف التجاري والذين يكسبون أكثر من 100000 ل. س في السنة
3. أسماء جميع الموظفين الذين يقطنون في نفس المدينة التي توجد فيها الشركة التي يعملون فيها.
4. أسماء جميع الموظفين الذين يقطنون في نفس المدينة والشارع الذين يقطن فيهما مدرائهم.
5. أسماء الشركات الموجودة في عدة مدن.
6. أسماء الشركات الموجودة في مدن للمصرف التجاري تواجد فيها.
7. أسماء الموظفين الذين رواتبهم أعلى من رواتب جميع موظفي المصرف التجاري.
8. حذف جميع الحدوديات المجودة في علاقة *works* والمتعلقة بموظفي المصرف التجاري.

الفصل السابع

لغة SQL

7 - 1 - المقدمة

لغة SQL هي تركيب من لغة الجبر العلاقاتي والحساب العلاقاتي. تجمع لغة SQL إمكانات إضافية إلى الاستعلام من قاعدة المعطيات. فتسمح بتعريف بنية المعطيات وإضافة وتعديل المعطيات في قاعدة المعطيات، وبتحديد أمانها.

تتكون لغة SQL من عدة أجزاء:

- **لغة تعريف المعطيات *DDL (Data Definition Language)*:** وتقدم التعليمات اللازمة لتعريف وتعديل مخطط علاقة، حذف علاقة، بناء فهارس .
- **لغة التعامل مع المعطيات *DML (Interactive Data Manipulation Language)*:** وتَعتمد طريقةً لصياغة الاستعلامات المرتكزة على الجبر العلاقاتي وجبر القضايا، وتحوي تعليمات لإضافة، وحذف وتعديل حدوديات في قاعدة المعطيات.
- **لغة التعامل مع المعطيات المضمَّنة *Embedded*:** وهي مصممة للتضمين في لغات البرمجة الاعتيادية مثل *PL/I*، باسكال، *C*، كوبول.
- **تعريف المنظار:** تحوي لغة تعريف المعطيات في *SQL* تعليمة تسمح بتعريف منظار.

- **السماحيات:** تحوي لغة تعريف المعطيات في *SQL* تعليمات لتحديد حقوق الوصول إلى العلاقات والمناظير.
- **التكامل:** تحوي *DDL* في *SQL* تعليمات لتحديد شروط التكامل.
- **التحكم في المناقلات *Transaction Control*:** في *SQL* يوجد تعليمات لتحديد بداية المناقلة ونهايتها. كما يوجد تطويرات تسمح بقفل المعطيات للتحكم في الوصول المتزامن.

في الأمثلة المقبلة سنستخدم قاعدة معطيات حول المصارف التي يتضمن مخططها المنطقي:

Branch_Schema = (branch_name, branch_city, assets)

Customer = (customer_name, customer_street, city)

Loan = (branch_name, loan_number, amount)

Borrower = (customer_name, loan_number)

Account = (branch_name, account_number, balance)

Depositor = (customer_name, account_number)

7 - 2 - لغة الاستعلام

تتألف البنية الأساسية للاستعلام في لغة *SQL* من ثلاثة أجزاء هي *Select, From, Where* . يُعبر الجزء *Select* عن عملية الإسقاط في الجبر العلاقاتي. و يُستخدم الجزء *From* لتحديد العلاقات المستخدمة في عملية الاختيار. ويُكافئ الجداء الديكارتي في الجبر العلاقاتي. أما الجزء *Where* فيتضمن قضية منطقية يجب أن تحققها الواصفات الموجودة في العلاقات التي جرى تحديدها في جزء *From*.

ويُصبح الشكل العام للاستعلام في لغة *SQL* كما يلي :

```
select A1, A2, ...... An
from   r1, r2, ...... rm
where  P
```

حيث: A_i هي واصفات

r_i علاقات

P قضية

وهي تكافىء في الجبر العلاقاتي العملية التالية :

$$\Pi_{A1-Am}\left(\sigma_P(r1\times r2\times....\times rm)\right)$$

- فقرة الاختيار *Select (Select clause)*: تُستخدم لتحديد قائمة الواصفات المطلوب إظهارها في نتيجة الاستعلام.

مثال: لإيجاد أسماء جميع الأفرع في علاقة القروض، نستخدم التعليمة :

```
select branch_name
From loan;
```

والنتيجة هي علاقة تحوي واصفاً واحداً *(branch_name)*

تسمح لغة *SQL* بتكرار لحدوديات في العلاقة أو في نتيجة استعلام، ولحذف التكرار نضيف كلمة *distinct* بعد *Select*.

```
Select distinct branch_name
From loan;
```

وتُستخدم كلمة *all* لمنع حذف التكرار في الحدوديات.

مثال :

```
select all  branch_name
from  loan;
```

يستخدم الرمز "*" للدلالة على اختيار جميع الواصفات من علاقة.

مثال :

```
select *
From loan;
```

ويمكن أن تحوي فقرة *select* تعابير حسابية تستخدم العمليات + ، - ، * و / .

مثال:

```
select branch_name, loan_number, amount * 100
from loan;
```

- **فقرة *Where*:** تُستخدم لتحديد الشروط التي يجب أن تحققها الحدوديات المختارة. فمثلاً لإيجاد أرقام القروض في الفرع *"perryridge"* والتي تزيد عن *1200* نكتب:

```
select loan_number
from loan
where branch_name = "perryridge" and amount >1200;
```

تستخدم المعاملات المنطقية *and*، *or* و *not* في فقرة *where* وعمليات المقارنة >، =>، =<، =و، <> و *between* و *not between*.

مثال :

```
Where amount between 90000 and 100000 ;
```

- **فقرة From:** تعرف فقرة *From* الجداء الديكارتي بين العلاقات المراد اختيار الحدوديات منها. ولما كانت عملية الدمج تُعرف كجداء ديكارتي وعملية اختيار وإسقاط، فإنه يمكننا التعبير بلغة *SQL* عن عملية الدمج التالية:

$$\Pi_{customer-name,loan-number}(borrower \rhd\lhd loan)$$

بالشكل:

```
select distinct customer_name, borrower. loan_number
From borrower, loan
where borrower.loan_number = loan.loan_number;
```

ويلاحظ في المثال السابق أننا استخدمنا *relation_name.attribute_name* في فقرة *select* بسبب وجود نفس اسم الواصف في أكثر من علاقة وذلك لتجنب الالتباس.

7 - 2 - 1 - إعادة التسمية

تجري إعادة التسمية للعلاقات والواصفات باستخدام الفقرة *as*:

old_name **as** *new_name*

مثال:

لإيجاد أسماء وأرقام قروض جميع الزبائن الذين حصلوا على قرض من فرع *Perryridge*، مع التعويض عن اسم العمود *loan_number* باسم *loan_id* نكتب:

```
select distinct customer_name, borrower.loan_number as
loan_id
from borrower, loan
where borrower.loan_number = loan.loan_number and
branch_name= "Perryridge"
```

7 - 2 - 2 - متحولات الحدودية

عبارة "*as*" مفيدة في تعريف متحولات من نمط حدودية. وكما في لغة القضايا فإن المتحول الحدودي في *SQL* يرتبط بعلاقة. لنبين ذلك بالمثال التالي:

مثال: لإيجاد جميع الزبائن الذين حصلوا على قرض من المصرف (أسماء الزبائن وأرقام قروضهم)

```
Select distinct customer_name, T.loan_number
From borrower as T, loan as S
Where T.loan_number = S.loan_number
```

يلاحظ من المثال السابق أن المتحول الحدودي *T* يرمز إلى حدودية لا على التعيين من العلاقة *borrower*. وفي بعض الحالات نحتاج إلى إعطاء تسميات مختلفة لمتحولات حدودية. كما يوضح ذلك المثال التالي:

لنفترض أننا نحتاج إلى معرفة جميع الفروع التي توجد في نفس المدينة التي يوجد فيها الفرع *Perryridge*. لحل هذا الاستعلام نستخدم متحولين الأول *S* والثاني *T* يشير كلاهما إلى العلاقة *Branch_Schema*.

Branch_name	Branch_city	Assets	
perryridge			← S
Y			← T

وتُستخدم التعليمة:

```
Select T.branch_name
From branch  T,
Branch S
Where S.city = T.city and S.branch_name= «perryridge»;
```

7 - 2 - 3 - العمليات على سلسلة المحارف

أكثر العمليات استخداماً هي التشابه الجزئي *like* ونصف هنا التشابه باستخدام حرفين:

% : للدلالة على أي سلسلة أحرف جزئية

under score: للدلالة على أي حرف

مثال:

للبحث عن الفروع التي تحوي سلسلة الأحرف *idge* في أي موقع من اسم الفرع نستخدم التعليمة:

```
Select  branch_name From Branch
Where branch_name Like «%idge%" ;
```

وتُستخدم "_ _ _" للدلالة على أي سلسلة أحرف مؤلفة من ثلاثة أحرف بالضبط

تسمح *SQL* باستخدام توابع مختلفة لسلسلة الأحرف مثل وصل سلسلتين *Concatenating* باستخدام *("II")*، واستخراج جزء من السلسلة، وإيجاد طول سلسلة الأحرف، والتحويل بين الأحرف الصغيرة والكبيرة، ...

7 - 2 - 4 - ترتيب النتائج

يمكن للمستثمر أن يتحكم في ترتيب الحدوديات في العلاقة الناتجة، وذلك باستخدام عبارة *Order by*. فمثلاً لاستخراج قائمة مرتبة ترتيباً أبجدياً بأسماء الزبائن، والذين حصلوا على قرض من الفرع *"Perryridge"* نكتب:

```
select distinct customer_name
from borrower, loan
where borrower.loan_number = loan.loan_number
        and branch_name = "perryridge"
        order by customer_name;
```

إن القائمة الناتجة ستكون مرتبة، ويمكن تحديد طريقة الترتيب تصاعدياً *asc* أو تنازلياً *desc* *(descending ascending)*. كما يمكن أن يطلب الترتيب على عدة واصفات.

مثال:

لنفترض أننا نريد قائمة القروض مرتبة ترتيباً تنازلياً حسب مبلغ القرض، وفي حال وجود عدة قروض لها نفس المبلغ، نقوم بترتيبها تصاعدياً حسب رقم القرض. نكتب:

```
select *
from loan
order by amount desc, loan_number asc
```

إن كلفة إجراء ترتيب على عدد كبير من الحدوديات هي كلفة عالية. ولذلك يجري إجراء الترتيب فقط في حال الضرورة.

7 - 2 - 5 - التوابع التجميعية

تُطبق التوابع التجميعية على مجموعة من القيم وتعيد قيمة واحدة. أهم هذه التوابع ما يلي:

التابع	دلالة التابع
Avg	المتوسط *Average*
Min	الأصغر *Minimum*
Max	الأكبر *Maximum*
Sum	المجموع *Total*

أمثلة:

- لإظهار متوسط أرصدة الحسابات في الفرع *"x"* نكتب :

```
select avg (balance)
from account
where branch_name = "x" ;
```

- ولإظهار أكبر قيمة قرض منحها فرع *Perryridge* نكتب :

```
select Max (amount)
from loan
where branch_name= «Perryridge»
```

- ولإظهار مجموع أرصدة الحسابات المفتوحة في الفرع *«x»* نكتب :

```
select Sum(balance)
from account
where branch_name =»x»;
```

- ولإظهار قائمة بالأفرع ومتوسط الأرصدة في كل منها نكتب :

```
select branch_name, avg (balance)
from account
group by branch_name ;
```

نلاحظ من المثال السابق أن الإبقاء على التكرار ضروري أثناء عملية حساب الوسطي *avg*.

7 - 2 - 6 - معالجة القيم غير المعلومة

تسمح لغة *SQL* باستخدام القيمة غير المعلومة *Null* للدلالة على عدم توفر معلومات في واصف.

مثال: لإيجاد أرقام القروض التي لا نعرف قيمتها نكتب:

```
select loan_number
from loan
where amount is null;
```

تعالج التوابع التجميعية القيم غير المعلومة بحيث تتجاهل وجود هذه القيم في مجموعة العناصر التي تُجرى عليها العملية التجميعية وتعطي النتيجة دون اعتبارها. ويقوم التابع *Count* بحصر عدد الحدوديات التي لاتحوي قيماً غير معلومة.

7 - 2 - 7 - تجزئة العلاقة

في الأمثلة السابقة كنا نطبق تابعاً تجميعياً على بعض أسطر العلاقة التي تحقق شرطاً معيناً مثل: "اسم الفرع=..." غير أننا لو أردنا مثلاً حساب وسطي الأرصدة في كل فرع فإننا، وبحسب الطريقة السابقة، سنضطر لتنفيذ تعليمة:

```
Select Avg (balance)
from account
Where branch_name= «x»;
```

عدداً من المرات يساوي عدد الفروع. وفي كل مرة نجعل x تأخذ اسم أحد الفروع.

طبعاً هذا الحل غير مجدٍ وخاصة عندما لا تكون لدينا فكرة سابقة عن أسماء كافة الفروع. لحل هذه المشكلة، تقدم لغة *SQL* إمكان تجزئة العلاقة وفق قيمة أحد الواصفات، ومن ثمّ إجراء عملية تجميع على واحد أو أكثر من الواصفات الباقية.

نحدد العمود الذي تجري التجزئة وفقه بالعبارة *Group by*. وهكذا نكتب:

```
select branch_name, avg(balance)
from account
group by branch_name;
```

الذي يمكن تمثيله بالشكل التالي :

Account

Branch_name	Account_nb	Balance
Perryridge	X1	10000
National	X2	20000
Perryridge	X3	30000
Perryridge	X4	20000
Perryridge	X5	40000

Results

Branch_name	Avg(balance)
Perryridge	25000
National	20000

مثال:

* لإيجاد عدد المشتركين الذين لديهم حساب في كل فرع نكتب:

```
select branch_name, count (distinct customer_name)
from depositor, account
where depositor.account_number =
 account.account_number
group by branch_name ;
```

ويمكن وضع شروط تُطبق على نتيجة عملية التجميع باعتبارها علاقة جديدة.

مثال:

* لإيجاد أسماء الفروع التي متوسط الأرصدة فيها أكبر من *1200$* يُستخدم التعبير *having* في *SQL* كالتالي:

```
select branch_name, avg (balance)
from account
group by branch_name
having avg (balance) > 1200;
```

إذا وجد تعبير *having* و *where* في نفس الاستعلام فإن الشرط *where* يطبق أولاً، وتوضع الحدوديات المحققة للشرط في مجموعات *(groups)* بتطبيق *group by* ثمّ يطبق التعبير *having* على كل مجموعة. وتكون النتيجة المجموعات المستخدمة في تعليمة *Select* والتي تحقق الاستفسار.

مثال:

*لإيجاد متوسط أرصدة كل زبون يعيش في مدينة "x" ويملك على الأقل ثلاثة حسابات نكتب:

```
select depositor, customer_name, avg (balance)
from depositor, account, customer
where depositor.account_number =
        account.account_number
and     depositor.customer_name =
customer.customer_ name
and customer_city = "x";
group by (depositor, customer_name)
having count (distinct depositor.account_number > = 3)
```

7 - 2 - 8 - الاستفسارات الجزئية المضمَّنة

تقدم *SQL* تقانات لتنفيذ استفسارات جزئية مضمَّنة. الاستفسار الجزئي هو تعبير من الشكل *Select -from -where* ويكون مضمناً في استفسار آخر.

أمثلة:

الانتماء إلى مجموعة

لإيجاد جميع الزبائن الذين لديهم قرض وحساب في المصرف. يمكننا الوصول إلى المطلوب باستخدام عملية التقاطع بين مجموعتين: مجموعة الزبائن المقترضين و مجموعة الزبائن المودعين. كما يمكن استخدام منحى آخر كإيجاد جميع المودعين في المصرف الذين ينتمون إلى مجموعة المقترضين من المصرف.

نكتب:

```
select distinct customer_name
from borrower
where customer_name in
(select customer_name from depositor)
```

مقارنة المجموعات

تسمح لغة *SQL* باستخدام المقارنة بين مجموعة حدوديات ومجموعة حدوديات أخرى.

مثال:

لإيجاد أسماء جميع الفروع التي قيم موجوداتها أكبر من قيم موجودات فرع واحد على الأقل من الفروع الموجودة في مدينة *"x"*.

يمكن صياغة الاستعلام بالشكل التالي:

```
select distinct T.branch_name
from branch as T, branch as S
where T.assets > S.assets and S.branch–city = "x"
```

تسمح *SQL* بالصياغة بشكل مختلف وباستخدام عبارة بعض *«some»* التي تدل على بعض عناصر المجموعة، والعبارة *all* التي تدل على كل عناصر المجموعة.

مثال:

```
select branch_name
from branch
where assets > some
(select assets
 from branch
 where branch–city = "*")
```

ويمكن استخدام عبارات المقارنة مع جزء من المجموعة التالية:

>some, <some, >=some, < >some, =some, <= some

وكذلك *>all, <all, >=all, < >all, =all , <=all*

مثال:

لإيجاد الفروع التي لديها متوسط الأرصدة أعظمي *(أي متوسط الأرصدة فيها أكبر من جميع متوسطات الأرصدة في باقي الفروع)* نكتب:

```
select branch_name
from account
group by branch_name
having avg (balance) >= all
(select avg (balance)
from account
group by branch_name)
```

7 - 2 - 9 - العلاقات المشتقة

يمكن حفظ نتيجة استفسار في علاقة جديدة. ويمكن إعادة تسمية واصفات هذه العلاقة الجديدة. وسنستخدم لذلك التعبير *as*:

مثال:

لإنشاء علاقة اسمها *result* تحوي الواصفين *branch_name* و *avg_ balance* نكتب:

```
(select branch_name, avg (balance)
from account
group by branch_name)
as result (branch_name, avg_balance)
```

7 - 3 - المناظير

يُعرّف المنظار بلغة SQL بالشكل:

```
create view as (query expression)
```

مثال:

```
create view branch_total_loan
(branch_name, total_loan)
as select branch_name, sum (amount)
from loan
group by branch_name
```

ولحذف منظار نكتب:

```
drop view view_name
```

7 - 4 - تعديل قاعدة المعطيات

يجري التعبير عن عمليات الإضافة والحذف والتعديل على قاعدة المعطيات باستخدام لغة SQL بالشكل التالي :

7 - 4 - 1 - الحذف

الشكل العام لتعليمة حذف حدوديات من علاقة هو:

```
delete From tablename where condition
```

أمثلة:

1- لحذف جميع الحسابات العائدة للفرع "Perryridge" نكتب:

```
delete from  account
where  branch_name= " Perryridge "
```

2- لحذف جميع الحسابات في جميع الفروع الموجودة في مدينة "Needham" نكتب:

```
delete from  account
where  branch_name in  (select  branch_name
from  branch
where  branch_city=  " Needham " )
```

3- لحذف كافة المودعين الذين فتحوا حسابات في فروع تقع في مدينة "Needham" نكتب:

```
delete from  depositor
where  account_number in
        (select  account_number
        from  branch, account
        where  branch_city=  " Needham "
        and
        branch.branch_name= account.branch_name
        )
```

4- لحذف جميع تسجيلات الحسابات التي رصيدها أقل من وسطي الأرصدة في المصرف نكتب:

```
delete from  account
where  balance <
        (select avg  ( balance) from  account)
```

المشكلة التي تظهر في هذا المثال هي أن حذف حدوديات من العلاقة *account* يغير من قيمة وسطي الأرصدة في المصرف، وهنا تقوم لغة *SQL* بحل هذا الإشكال كالتالي:

في البداية تقوم بحساب متوسط الأرصدة وإيجاد جميع الحدوديات التي يجب أن تُحذف.

ثُم تقوم بحذف جميع الحدوديات الموجودة سابقاً (دون إعادة حساب المتوسط أو إعادة اختبار الحدوديات).

7 - 4 - 2- الإضافة

تأخذ عملية إضافة حدودية إلى علاقة الشكل العام التالي :

```
insert into  rel_name
values  ( attribute values)
```

أو

```
insert into  rel_name
```

(.... select ....From....Where)

وفي هذه الحالة يجري تنفيذ عملية الاختيار أولاً، ثمّ عملية الإضافة.

أمثلة:

1- لإضافة حدودية جديدة إلى علاقة الحسابات نكتب:

```
insert into  account
values  ( " Perryridge " , A_9732, 1200)
```

أو الشكل المكافئ:

```
insert into  account
        ( branch_name, balance, account_number)
values
        ( " Perryridge " , 1200, A_9732)
```

2- لإضافة حدودية جديدة إلى علاقة الحسابات مع أن الرصيد للحساب مجهول أو غير معلوم نكتب:

```
insert into  account
values  ( " Perryridge " , A_777, null)
```

3- لإضافة حساب مصرفي إلى جميع المقترضين من المصرف من الفرع *"Perryridge"* بقيمة *200$* وبحيث يُعتمد رقم القرض رقماً للحساب الجديد، نكتب:

```
insert into  account
select  branch_name, loan_number, 200
from  loan
where  branch_name=  " Perryridge "
```

```
insert into  depositor
select  customer_name, loan_number
from  loan, borrower
where  branch_name=  " Perryridge "
and  loan.account_number= borrower.account_number
```

7 - 4 - 3 - التعديل

الشكل العام لتعليمة التعديل هو:

```
update rel_name

set attribute= new_values

where condition
```

مثال:

لتعديل أرصدة الحسابات المصرفية بإضافة 6% إلى الحسابات التي يزيد رصيدها عن على 10.000$ وإضافة 5% إلى البقية، نكتب:

```
update account

set balance = balance * 1.06

where balance> 10000
```

```
update account

set balance = balance* 1.05

where balance <= 10000
```

ويمكن إجراء التعديل على قاعدة المعطيات بواسطة التعديل على المنظار المعرف على هذه القاعدة. ولكن كما ناقشنا سابقاً (الفصل الثالث) تظهر بعض المشاكل المتعلقة بالتعامل مع القيم غير المعلومة.

مثال:

ليكن لدينا المنظار *branch_loan* المُعرف على قاعدة معطيات المصرف. والذي يُظهر معطيات جميع القروض مع إخفاء المعطيات المتعلقة بكمية هذه القروض.

```
reate view  branch_loan as

select  branch_name, loan_number

from  loan
```

نضيف حدودية جديدة إلى المنظار:

```
insert into  branch_loan

values  ( " Perryridge " ,  " L_307 " )
```

تتمثل عملية الإضافة هذه بعملية إضافة للحدودية

("Perryridge", "L_307" , null)

إلى العلاقة *loan*.

التعديل على المعطيات من خلال مناظير أكثر تعقيداً. وفي بعض الأحيان تستحيل ترجمته لمعرفة التعديلات الواجب إجراؤها على العلاقات الأساسية.

7 - 5 - لغة تعريف المعطيات DDL

7 - 5 - 1 - تعريف العلاقات

تسمح لغة *SQL* بتعريف العلاقات التي تتكون منها قاعدة المعطيات. وبتحديد معلومات عن كل علاقة بما يتضمن:

- المخطط العلاقاتي لكل علاقة.
- مجال تعريف القيم المرتبطة بكل واصف.
- شروط التكامل.
- مجموعة المؤشرات المرتبطة بكل علاقة.
- أمن المعلومات وسماحيات الوصول.
- بنية التخزين الفيزيائي.

يُعرّف مخطط علاقة بلغة *SQL* باستخدام التعليمة التالية:

```
create table  r ( A1 D1 ,A2 D2 , ..., An Dn ,
integrity-constraint l i  ,
...,
integrity-constraint k i  )
```

حيث :

r اسم العلاقة

Ai اسم الواصفة رقم *I* من مخطط العلاقة *r*

Di مجال تعريف الواصفة *Ai*

li, …,ki شروط تكامل مُعرّف على الجدول

مثال:

```
create table  branch
 ( branch_name char(15) not null ,
 branch_city char(30),
 assets integer)
```

تُعرّف شروط التكامل على المخطط العلاقاتي والمتضمنة:

- عدم احتوائه على قيم غير معلومة *not null*.
- تعريف المفتاح الرئيسي *primary key (A1 ,...,An)*
- تعريف قضية *check (P)* حيث *P* قضية.

مثال:

لتعريف مخطط علاقاتي *branch* بحيث تكون الواصفة *branch_name* مفتاحاً رئيسياً و قيم الواصفة *assets* غير سالبة، نكتب:

```
create table branch
(
branch_name char(15) not null,
branch_city char(30),
assets integer,
primary key ( branch_name), check ( assets>=0)
)
```

إن تعريف المفتاح الرئيسي على واصفة يجعل اختبار عدم احتوائها على قيم غير معلومة آلياً.

7 - 5 - 2- حذف مخطط علاقة

تسمح تعليمة *drop table* بحذف جميع المعلومات المتعلقة بالعلاقة من قاعدة المعطيات.

7 - 5 - 3 - تعديل مخطط علاقة

تسمح تعليمة *alter table* بإضافة واصفات إلى مخطط علاقة موجودة، وبحيث تأخذ هذه الواصفة قيمة غير معلومة في جميع

الحدوديات الموجودة سابقاً في العلاقة.

مثال:

لإضافة واصف باسم *A* ومجاله *D* إلى العلاقة *r* نكتب:

```
alter table r add A D
```

ويمكن استخدام تعليمة *alter table* لحذف واصف من العلاقة.

مثال:

```
alter table r drop A
```

حيث *A* واصف ضمن العلاقة *r*.

7 - 6 - لغة *SQL* المضمنة

يمكن تضمين تعليمات *SQL* في لغات برمجة متعددة مثل: *Pascal, PL/I, Fortran, C* و*Cobol*. تُسمى اللغة المُتضمنة تعليمات *SQL* باللغة المُضيفة.

الشكل العام لتعليمات *SQL* المضمنة في لغة برمجة مثل *PL / I* هو:

```
EXEC SQL
        < embedded SQL statement >
END EXEC
```

مثال:

لإيجاد أسماء وأرقام الحسابات المصرفية للزبائن الذين يزيد رصيد حساباتهم عن *amount*، نقوم بتحديد تعليمة *SQL* التي تسمح باستخراج تلك المعلومات وبتعريف مؤشر *cursor* لتلك التعليمة

كالتالي:

```
EXEC SQL
declare c cursor for
select customer_name, account_number
from depositor, account
where depositor.account_number= account.account_number
and account.balance> : amount
```

- تسمح تعليمة *open c* حيث *c* مؤشر تعليمة *SQL* بوضع التعليمة موضع التنفيذ.

```
EXEC SQL
  open  c
```

- تسمح تعليمة *fetch c into :variable-name* بإسناد قيم واصفات حدودية واحدة من نتيجة الاستعلام إلى متحولات في اللغة المضيفة.

```
EXEC SQL
fetch  c into  :cn, :an
```

ويُحصل على الحدوديات المختلفة الناتجة من استعلام معين، باستدعاء متتالٍ لتعليمة *fetch*، ويستخدم متحول خاص لمنطقة اتصال لغة *SQL* للإشارة إلى الوصول إلى نهاية نتيجة الاستعلام.

- تسمح تعليمة *close c* لنظام قواعد المعطيات بحذف العلاقة الوسيطة الحاوية لنتيجة الاستعلام.

```
EXEC SQL
close  c
END-EXEC
```

7 - 7 - تمارين

1- أعد كل طلبات المبينة في نهاية الفصل السابق مع صياغة المطلوب بلغة *SQL*.

2- لتكن لدينا العلاقات التالية:

employee (Id, name, job, manager_id, salary, dept_id)

department (dept_id, name, city)

التي تتضمن معلومات عن الموظفين في شركة تمتلك عدة فروع.

حيث:

employee

Id	رقم الموظف
name	اسم الموظف
job	العمل الذي يؤديه الموظف
manager_id	رقم الرئيس المباشر
salary	الراتب الشهري
dept_id	رقم القسم الذي يعمل فيه

department

dept_id	رقم القسم
name	اسم القسم
city	المدينة التي يوجد فيها

المطلوب:

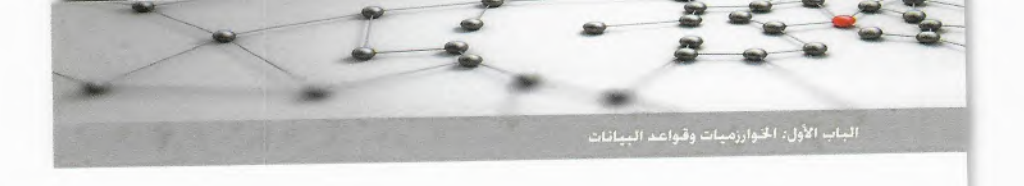
1. اكتب التعليمات اللازمة لإنشاء هاتين العلاقتين مع تحديد الشروط التالية:

- رقم الموظف وحيد
- لايمكن أن يكون اسم الموظف مجهولاً
- يعمل كل موظف في قسم واحد

2. هل يمكن اعتبار رقم الرئيس المباشر إجبارياً

3. اكتب الاستفسارات التالية بلغة SQL:

- عدد العاملين في مدينة دمشق مرتبة أبجدياً
- أسماء ورواتب العاملين في قسم التسويق (Marketing) مرتبة تنازلياً وفق الراتب
- اسم المدير العام للشركة
- الموظفون الذين يتبعون مباشرة إلى المدير العام
- الموظف (أو الموظفون) الذي يحصل على أعلى راتب في الشركة
- أعداد العاملين في كل قسم من أقسام الشركة
- اسم القسم الذي يحوي أكبر عدد من العاملين
- اسم القسم الذي يزيد مجموع رواتب العاملين فيه عن 100000 ل.س
- أسماء ورواتب العاملين الذين يزيد راتبهم عن راتب رئيسهم المباشر
- القسم الذي يزيد متوسط رواتب العاملين عن متوسط رواتب العاملين في الشركة



الباب الثاني

البرمجيات

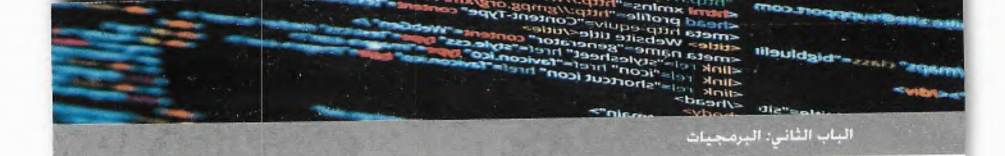
(Software)

```
Any time,
<!DOCTYPE html PUBLI
"http://www w3 org/TR/xh
<html xmlns="http://www.
<head profile="http://gmp
<meta http-equiv="Conten
<title> Website title</title>
<meta name="generator"
<link rel="stylesheet" href=
<link rel="icon" href="favic
<link rel="shortcut icon" h
</head>
<body>
<div id="setmain">
<div class="topdiv">
<div id="flashlogo">
```

الفصل الأول

الأوتومات واللغات الصورية والمترجمات

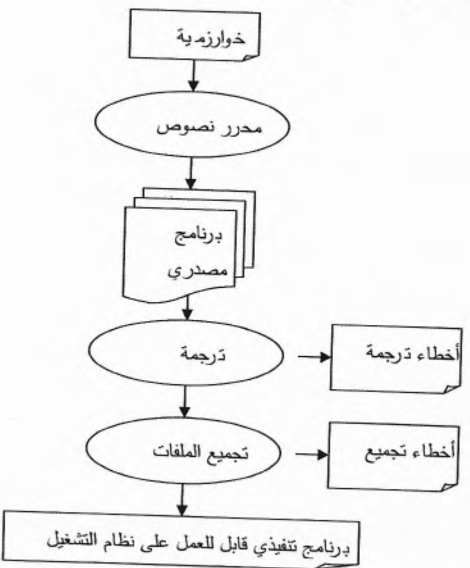
- المقدمة
- بنية مترجم
- التحليل المفرداتي
- اللغات الصورية والأوتومات
- التحليل القواعدي
- التحليل الدلالي
- توليد الرماز



1 - 1 - المقدمة

يستخدم أي مبرمج أداة ضرورية جداً في عملية البرمجة، ندعوها المترجم. يمكننا تعريف المترجم بأنه برنامج حاسوبي يترجم النص البرمجي الذي نكتبه بلغة برمجة عالية المستوى *(C, Pascal, C++, C#, Java, ...etc)* إلى مجموعة تعليمات قابلة للتنفيذ من قبل الحاسوب. تكون هذه التعليمات التنفيذية مكتوبة بلغة منخفضة المستوى سواء كانت لغة ثنائية مؤلفة من أصفار ووحدان *(0,1)*، أو لغة تجميع. ويمكن أيضاً بناء مترجمات من لغة عالية المستوى إلى لغة أخرى عالية المستوى وخصوصاً عندما تكون هناك ضرورة لنقل رماز *(كود)* برمجي من بيئة برمجية إلى أخرى.

عموماً تمر عملية تشغيل برنامج حاسوبي بمجموعة من المراحل التي نمثلها في الشكل التالي:



تجري كتابة النص البرمجي (أو النصوص البرمجية) لأي برنامج حاسوبي، باستخدام محرر نصوص (ضمن ملف واحد أو ضمن مجموعة من الملفات). ندعو هذه النصوص البرمجية التي تؤلف برنامج حاسوبي، بالبرنامج المصدري (*Source Program*).

يمر البرنامج المصدري بعد ذلك، بمرحلة ترجمة (*Compilation*) وتجميع (*Linking*) يجري فيها ربط الملفات الحاوية على البرنامج المصدري الواحد ببعضها البعض وترجمتها إلى مجموعة من التعليمات التنفيذية المكتوبة بلغة منخفضة المستوى.

تشكل التعليمات التنفيذية الناتجة برنامج جديد ندعوه البرنامج الهدف (*Destination Program*). يأخذ في الحالة العامة شكل ملف تنفيذي قابل للتشغيل مباشرةً على نظام التشغيل.

تظهر خلال مراحل الترجمة والتجميع أخطاء ندعوها أخطاء الترجمة (تكون ناجمة عن أخطاء في نصوص البرنامج المصدري) أو أخطاء التجميع (تكون ناجمة عن أخطاء في ربط الملفات الحاوية على النصوص). تؤدي هذه الأخطاء إلى توقف عملية الترجمة حتى يجري تصحيحها من قبل المبرمج، قبل إعادة تشغيل المترجم لتوليد "برنامج هدف" خال من الأخطاء.

تجدر الإشارة إلى أن عملية بناء المترجم (والذي عرّفناه كبرنامج حاسوبي) تتعلق بعنصرين أثنين بآن واحد:

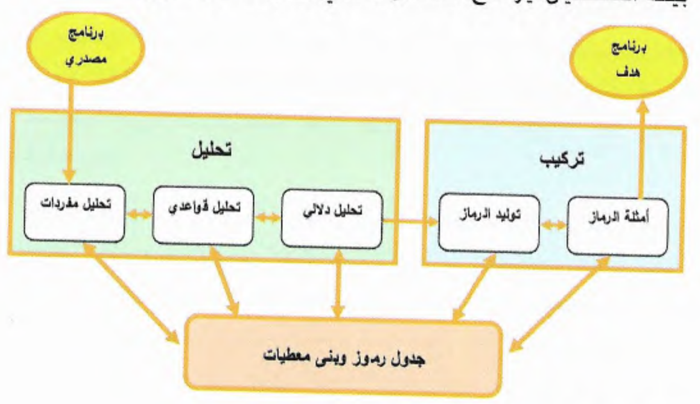
1- لغة البرمجة المصدرية عالية المستوى الذي يستخدمها المبرمج.

2- نظام التشغيل أو البيئة التي سيجري تشغيل البرنامج عليها.

فعلى سبيل المثال، يختلف مترجم لغة*C++* الذي يعمل على نظام *Windows* عن مترجم لغة *C++* الذي يعمل على نظام *Linux*، نظراً لضرورة توليد تعليمات تشغيل تنفيذية مختلفة في هاتين الحالتين، بالرغم من أننا نتكلم عن نفس اللغة وهي *C++*. في حين، يختلف مترجم لغة *C++* عن مترجم لغة *Java* حتى ولو كان المترجمان يعملان على نظام *Windows*، نظراً لأننا نترجم لغتين برمجيتين مختلفتين.

1 - 2 - بنية مترجم

تتألف عملية الترجمة من مرحلتين أساسيتين: مرحلة التحليل والتي يجري فيها تقسيم النص البرمجي إلى كلمات وجمل والتأكد من صحتها ودلالاتها، ومرحلة التركيب التي يجري فيها تركيب نص برمجي جديد بنفس دلالة النص المصدري ولكن بلغة أخرى هي عموماً اللغة التي تتألف منها التعليمات التنفيذية التي يفهمها نظام أو بيئة التشغيل. يوضح الشكل التالي مكونات المرحلتين:



1 - 2 - 1 - مرحلة التحليل

التحليل المفرداتي *(Lexical Analysis)*

تجري في هذه المرحلة عملية التعرف على أنواع الكلمات المؤلفة للنص البرمجي المصدري. لذا تجري قراءة النص البرمجي المصدري حرفاً حرفاً، وتجميع الحروف لتشكيل كلمات. يتولى التحليل المفرداتي المهام التالية:

1. حذف كافة المحارف التي لا تدخل في صلب النص البرمجي مثل الفراغات، التعليقات، ... الخ.

2. تجميع المحارف في كلمات وتحديد نوع كل كلمة: كلمة مفتاحية من لغة البرمجة، متحول *(Variable)*، ثابت *(Constant)*، قيمة عددية *(Numeric Value)*، عملية حسابية، عملية منطقية، ... الخ.

التحليل القواعدي (Syntax Analysis)

يمكن أيضاً تسميته بالتحليل الصرفي. إذ يجري خلال التحليل القواعدي جمع الكلمات الناتجة عن التحليل المفرداتي، في جمل وبنى قواعدية تشكل بنية النص البرمجي. يقوم المحلل القواعدي بالتأكد من سلامة الجمل المبنية بالنسبة لقواعد صرفية خاصة بكل لغة برمجة. فعلى سبيل المثال، تحدد القواعد الصرفية طريقة بناء الجملة الشرطية في لغة البرمجة *(مثل لغة C++)*، أو طريقة بناء حلقة تكرار فيها.

تشبه عملية التحليل القواعدي للغة البرمجة عملية الإعراب في اللغات الطبيعية. إذ يمكن لجملة عربية أن تكون فعلية مؤلفة من فعل وفاعل ومفعول به، أو اسمية مؤلفة من مبتدأ وخبر. كذلك هو الحال في جملة شرطية مكتوبة بلغة برمجة مثل لغة *C*. إذ يجب أن تبدأ هذه الجملة بكلمة *"if"* يليها تعبير شرطي يعبر عن شرط معين مثل *"X>0"* ومن ثم مجموعة من العمليات التي يجب تنفيذها عند تحقق الشرط، ويمكن أن تتبع هذه العمليات كلمة *"else"* لتعريف العمليات التي تجري عند انتفاء الشرط، أو أن تنتهي الجملة الشرطية دون كلمة *"else"*.

التحليل الدلالي (Semantic Analysis)

تجري في هذه المرحلة عملية التحقق من دلالة الجمل المركبة بعد أن تم التأكد من سلامتها قواعدياً. فعلى سبيل المثال، تعتبر عملية التحقق من الأنماط، مرحلة أساسية من مراحل التحليل الدلالي، حيث يجري فيها التأكد من أن عملية مثل *(X * Y)* لها دلالة إذا كانت كل من *X*

و Y تعبر عن قيمتين رقميتين، في حين تصبح هذه العملية دون دلالة (إلا في حالات خاصة ليست في صلب دراستنا هنا) في حال كانت X تعبر عن قيمة رقمية و Y تعبر عن سلسلة محارف.

1 - 2 - 2 - مرحلة التركيب والتوليد

توليد الرماز

تجري فيها عملية توليد التعليمات التنفيذية التي لها نفس دلالة تعليمات النص البرمجي المصدري ولكن بلغة مفهومة من كلٍ من بيئة ونظام التشغيل اللذين سيجري تشغيل البرنامج عليهما. يمكن في بعض الأحيان توليد الرماز بلغة أخرى عالية المستوى (في حال كان المطلوب هو نقل نصوص برمجية مكتوبة بلغة برمجة قديمة إلى نصوص برمجية مكتوبة بلغة برمجة أحدث لتسهيل الربط مع برامج أخرى). كما يمكن توليد الرماز بلغة خاصة بآلة افتراضية تعمل على نظام تشغيل كما هو الحالة في آلة *Java* الافتراضية (*Java Virtual Machine*) التي تعمل على نظام التشغيل ويندوز أو غيره. أو بلغة آلة خاصة بنوع معين من المعالجات وضمن بيئة نظام تشغيل كما هو الحال في بيئة التشغيل *Windows* التي تعمل على معالجات الحواسب الشخصية *X86*.

أمثلة الرماز

تهتم هذه المرحلة باختصار وتحسين الرماز المولد وذلك بهدف تسريع عمل البرنامج التنفيذي الناتج وضمان تنفيذ سريع وفعال له عند تشغيله. يجري في هذه المرحلة حذف تعليمات لا معنى لها مثل تعريف متحول وعدم استخدامه، ضرب قيمة ما بـ *1*، جمع قيمة ما مع *0* ... الخ.

1 - 2 - 3 - مراحل موازية

إدارة جدول الرموز (*Symbol Table*)

يشكل جدول الرموز أحد أهم بنى المعطيات المستخدمة في المترجمات. إذ يجري تخزين المتحولات المعرفة ضمن النص البرمجي المصدري في جدول الرموز. كما يجري تخزين أنماطها التي جرى الإعلان عنها في النص البرمجي.

يُستخدم جدول الرموز أيضاً لتحديد مجال رؤية أو مدى المتحول، وهو مجموعة مقاطع النص البرمجي وإجرائياته التي يمكن فيها استخدام هذا المتحول وفقاً لتعريفه في جدول الرموز. فعلى سبيل المثال، في بعض لغات البرمجة وعند الإعلان عن المتحول *X* كمتحول محلي من النمط *Integer* ضمن الإجرائية *Proc*، لا يمكن للمبرمج استخدام هذا المتحول ضمن النص البرمجي خارج نطاق الإجرائية *Proc*. لذا يقوم المترجم بالاعتماد على جدول الرموز لتخزين اسم المتحول ونمطه ومداه وذلك بهدف التأكد من عدم استخدام هذا المتحول خارج *Proc*.

إدارة الأخطاء (*Error Handling*)

تنتج عن عمليات التحليل والتركيب السابقة أخطاء ارتكبها المبرمج أثناء كتابته للنص البرمجي: منها أخطاء في كتابة الكلمات، ومنها أخطاء في بناء الجمل، وبعضها أخطاء دلالية ... وهكذا دواليك.

يجري التعامل مع كل نوع من أنواع الأخطاء بشكل مختلف ولكن يبقى الهدف الأول والأخير لمعالجة الأخطاء هو إعطاء المبرمج إشارة إلى الخطأ، وتوضيح سبب الخطأ (قدر الإمكان)، ومحاولة تجميع أكبر عدد من الأخطاء الناجمة عن خطأ أول وإظهارها بآن واحد وذلك بهدف تسريع عملية الترجمة.

1 - 3 - التحليل المفرداتي (Lexical Analysis)

يشكل المحلل المفرداتي الجزء الأول من المترجم وينفذ المرحلة الأولى من عملية الترجمة. تتلخص المهمة الأساسية للمحلل اللفظي بتجميع محارف الدخل، الآتية من النص البرمجي المصدري، بهدف توليد مجموعة من الكلمات التي ستؤلف بدورها جمل يعالجها المحلل القواعدي.

ينفذ المحلل المفرداتي أيضاً مجموعة من المهام الثانوية من أهمها: حذف المحارف التي لا دور لها: كالفراغ وسلاسل التعليقات. كما يتولى مهمة حفظ أرقام أسطر النص البرمجي التي يمكن الاستعانة بها للإشارة إلى مكان الأخطاء الموجودة في النص البرمجي.

سنستعرض في هذا القسم بنية المحلل المفرداتي وتفاصيل عملية التحليل المفرداتي. تجدر الإشارة إلى أن مسألة تصميم وتشغيل محلل مفرداتي تكافئ مسألة تصميم برنامج ينفذ عمليات على سلاسل من المحارف ويهتم بالتعرف على أشكال وصيغ محددة لهذه السلاسل، وهي مسألة تصب في مجال بناء المنظومات التي تساعد في البحث عن المعلومات وتشكل صلب محركات البحث.

1 - 3 - 1 - المفردات

في كل نص برمجي تشكل كل مجموعة من الأحرف المتتالية، "**كلمة**". يكون لكل كلمة من الكلمات المستخدمة في النص البرمجي شكل يحدد انتماءها إلى نوع من الكلمات أو ما ندعوه "**نموذج**" ***(Model)***. ونستخدم لمجموعة الكلمات التي لها نموذج محدد إسم ندعوه "**مفردة**" ***(Token)***.

فعلى سبيل المثال: تعبر الكلمات: *(+ ، - ، * ، / ، %)* عن عمليات حسابية. يمكن أن نستخدم للتعبير عن العمليات الحسابية المفردة *ARTHOP (من Arithmetic Operator)*.

كما تعبر الكلمات *(X, Counter, Y, Toto)* عن متحولات يستخدمها المبرمج. ويمكن أن نستخدم المفردة *IDENT* *(من Identifier)* للتعبير عن المتحولات. كما يكون للمتحولات نموذج يعبر عن شكل المتحول وهو ما سنراه لاحقاً (فعلى سبيل المثال يفرض النموذج الذي يعرِّف شكل المتحولات عدم استخدام أرقام لتعريف متحول. إذ لا يمكن للمحلل المفرداتي اعتبار *55* متحول. فالمحلل يعتبر *55* قيمة رقمية).

1 - 3 - 2 - التعابير المنتظمة

نستعرض في هذه الفقرة مجموعة من التعاريف والمبادئ الخاصة بنظرية اللغات. وهي تعاريف ومبادئ مُستخدمة بكثافة في عملية التحليل المفرداتي.

> **تعريف:** ندعو "**أبجدية**" *(Alphabet)*. مجموعة منتهية غير خالية Σ من الرموز. كما ندعو "**كلمة**" *(Word)* كل سلسلة
>
> منتهية من عناصر Σ.

نستخدم الرمز ε للدلالة على الكلمة الفارغة. كما نستخدم الرمز Σ^* للدلالة على المجموعة غير المنتهية التي تضم جميع الكلمات الممكنة المبنية اعتباراً من الأبجدية Σ. ونستخدم الرمز Σ^+ للدلالة على مجموعة الكلمات غير الفارغة التي يمكن أن نبنيها اعتباراً من Σ.

بالنتيجة يكون: $\Sigma^+ = \Sigma^* - \{\varepsilon\}$.

يشير الرمز $|m|$ إلى طول الكلمة *m*. ويعبر عن عدد الأحرف المنتمية إلى الأبجدية Σ والتي تشكل الكلمة *m*. ونستخدم الرمز Σ^n للدلالة

على مجموعة الكلمات المنتمية إلى Σ^* والتي طولها *n*. بالتالي يمكننا أن نستنتج أن: $\Sigma^* = \sum_{n=0}^{\infty} \Sigma^n$.

تعريف: نعرف عملية "الدمج التسلسلي" أو "التتابع" *(Concatenation)* والتي نرمز لها بالرمز " • ". بأنها عملية نطبقها على كلمتين كالتالي:

لتكن لدينا الكلمة $u = u_1 ... u_n$ (حيث $u_i \in \Sigma^*$) والكلمة

$v = v_1 ... v_p$ (حيث $v_i \in \Sigma^*$). يكون حاصل الدمج

التسلسلي للكلمتين *u* و *v*. الكلمة $u \bullet v = u_1 ... u_n v_1 ... v_p$

. بشكل عام يمكن إهمال رمز الدمج التسلسلي " • " وكتابة *uv* بدلاً عن $u \bullet v$.

فعلى سبيل المثال. لنأخذ الأبجدية $\Sigma = \{a, b, c\}$. والكلمات *u*، *v*، *w*، *t*. حيث *u=aaba*، *v=bbbacbb*، *w=c*، $t = \varepsilon$. هي كلمات تنتمي إلى Σ^* وبأطوال: *4*، *7*، *1*، *0* على التسلسل. تكون الكلمة $u \bullet v = aababbbacbb$ حاصل الدمج التسلسلي للكلمتين *u* و *v*.

خصائص:

✓ $|u \bullet v| = |u| + |v|$

✓ $(u \bullet v) \bullet w = u \bullet (v \bullet w)$

✓ $u \bullet \varepsilon = \varepsilon \bullet u = u$

تعريف: ندعو لغة مبنية على أبجدية Σ، كل مجموعة جزئية من Σ^*.

على سبيل المثال، يمكن اعتباراً من الأبجدية $\Sigma = \{a, b, c\}$ تعريف لغة غير منتهية L_1 حيث تتألف هذه اللغة من الكلمات:

$\{ \varepsilon , a, b, aab, bbcaa, bbcccacccaaaabbbcaa, \dots \}$

تعريف: العمليات على اللغات

الاجتماع:

$$L_1 \cup L_2 = \{w : w \in L_1 \quad OR \quad w \in L_2\}$$

التقاطع:

$$L_1 \cap L_2 = \{w : w \in L_1 \quad AND \quad w \in L_2\}$$

الدمج التسلسلي أو التتابع:

$$L_1 L_2 = \{w = w_1 w_2 : w_1 \in L_1 \quad AND \quad w_2 \in L_2\}$$

الإغلاق:

$$L^* = \bigcup_{n \geq 0} L^n$$

السؤال الذي نطرحه: في حال كان لدينا لغة ما L. كيف يمكن توصيف الكلمات المنتمية إلى اللغة؟ وكيف يمكن توصيف هذه اللغة. عموماً، توجد عدة أنماط من اللغات وهو ما سنراه في فصل لاحق، ولكن سنركز في هذا الفصل على اللغات التي ندعوها لغات منتظمة.

تعريف: نستطيع تعريف "**لغة منتظمة**" L *(Regular Language)* على أبجدية Σ بشكل عودي كما يلي:

- ✓ $\{\varepsilon\}$ هي لغة منتظمة على Σ.
- ✓ إذا كان a حرف من حروف الأبجدية Σ، تكون $\{a\}$ لغة منتظمة على Σ.
- ✓ إذا كانت L لغة منتظمة على Σ، تكون كلٌّ من L^n حيث $n \geq 0$، و L^* لغات منتظمة على Σ.
- ✓ إذا كانت L لغة منتظمة على Σ، تكون متممة L بالنسبة للمجموعة الكلية Σ^* ونرمز لها $\overline{L}$ هي لغة منتظمة.
- ✓ إذا كان كلٌّ من L_1 و L_2 لغات منتظمة على Σ، تكون كلٌّ من $L_1 \cup L_2$ و L_1L_2 و $L_1 \cap L_2$ لغات منتظمة على Σ.

تعريف: يمكن توصيف اللغات منتظمة، باستخدام أداة ندعوها **"التعابير المنتظمة"** *(Regular Expressions)*. نعطي فيما يلي تعريف عودي للتعابير المنتظمة:

- ✓ ε هو تعبير منتظم يوصف اللغة $\{\varepsilon\}$.
- ✓ إذا كان a حرف من حروف الأبجدية Σ، يكون a تعبير منتظم يوصف اللغة $\{a\}$.
- ✓ إذا كان r تعبير منتظم يوصف اللغة L فإن كلاً من $(r)^*$ و $(r)^+$ عبارة عن تعبيرين منتظمين يوصفان اللغتين L^* و L^+ على الترتيب.
- ✓ إذا كان r_1 و r_2 تعبيرين منتظمين يوصفان اللغتين L_1 و L_2 على الترتيب، فإن كلاً من $r + r_2$ و $r_1 r_2$ عبارة عن تعبيرين منتظمين يوصفان اللغتين $L_1 \cup L_2$ و $L_1 L_2$ على الترتيب.

أمثلة عن التعابير المنتظمة

- ✓ يوصف $(a+b)^*$ مجموعة الكلمات *(اللغة)* المؤلفة من a و b، أو الكلمة الفارغة.
- ✓ يكون $(b+a)^* = (a+b)^*$
- ✓ يوصف $(a+b)^* bbb (a+b)^*$ مجموعة الكلمات *(اللغة)* التي تحتوي على السلسلة bbb فيها.

1 - 3 - 3 - تنفيذ التحليل المفرداتي

a. توصيف المفردات

توفر لنا التعابير المنتظمة أداة لوصف الكلمات التي لها نموذج محدد. يتم التعبير عن هذا النموذج بمفردة. فعلى سبيل المثال، جميعنا يعلم أن استخدام اسم أي متحول في نص برمجي يخضع لقواعد محددة:

- ✓ أن يبدأ الاسم بحرف ومن ثم نستخدم أي حرف، أو رقم، بالإضافة إلى الحرف "_" *(underscore)*، ودون أن يكون هناك حدود لطول اسم المتحول يمكن استخدام الأحرف الصغيرة *(small letters)* أو الكبيرة *(capital letters)*.

وعليه يمكننا أن نعرّف المفردة *IDENT* المعبرة عن المتحولات، وذلك اعتماداً على تعبير منتظم، كما يلي:

$IDENT=(a|b|c \quad ...|y|z|A|B|C....|Y|Z) \quad (a|b|c \quad ...|y|z|A|B|C...|Y|Z|0|1|2|3|4|5|6|7|8|9)*""$

لاختصار العمليات يمكننا أن نعرف ما يلي:

$Letter = a - z \mid A - Z$

$Digit = 0 - 9$

$Sep = _$

$EndOfString=""$

وبالتالي يصبح تعريف *IDENT* على الشكل التالي:

$IDENT = Letter\ (Letter \mid Digit \mid Sep)*\ EndOfString$

تجدر الإشارة إلى عدم إمكانية تمثيل كل نماذج الكلمات (وبالتالي عدم إمكانية تعريف المفردات المعبرة عنها) باستخدام التعابير المنتظمة. فعلى سبيل المثال، في حال كان لدينا في لغة معرفة على أبجدية *{a,b}* نموذج كلمات له الشكل $a^n b^n$ يتبع القاعدة التالية: "تبدأ

الكلمة بالحرف *a* الذي يتكرر *n* مرة ومن ثم يتبعها نفس العدد من *b* الذي سيظهر *n* مرة أيضاً من أجل أي *n*). فإننا ببساطة لا نستطيع استخدام التعابير المنتظمة للتعبير عن هذا النموذج (سيتم توضيح السبب في فصل لاحق).

b. تحديد الرموز

تنقسم رموز لغة برمجة إلى:

- ✓ الكلمات المفتاحية الأساسية مثل *if*، *while*، ... بالإضافة إلى الرموز البسيطة كالعلاقات الحسابية *(+، -، *، /، %)* أو علاقات المقارنة *(<، >، =<، =>)* وغيرها، فإننا نكتفي بتعريف المفردة المعبرة عنها مثل *ARTHOP* التي تعبر عن العلاقات الحسابية.

- ✓ بالنسبة للرموز التي تحتاج لقواعد محددة لكتابتها (لها نموذج) مثل المتحولات، فإننا (وكما أوضحنا في الفقرة السابقة) نحدد قواعد تعريف نموذجها ونعطي تعريف للمفردات المعبرة عنها كما هو الحال مع *IDENT*. بنفس الطريقة يمكن أن نستخدم المفردة *REAL* للتعبير عن عدد حقيقي (مثل *5.3*) والذي تكون قاعدة كتابته كما يلي:

POINT = "."

REAL = (Digit) POINT (Digit)**

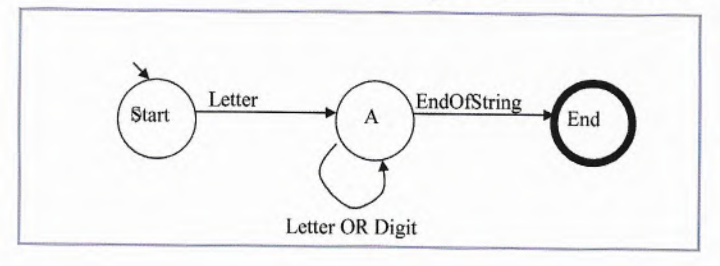
c. تحليل كلمات النص البرمجي

لنفترض أننا نحاول كتابة برنامح للتعرف على أسماء المتحولات بحيث يأخذ البرنامج في دخله اسم ويعيد قيمة *1* أو *0* للدلالة على أن الكلمة تعبر عن اسم متحول أو لا تعبر على الترتيب. لنفترض أن اسم المتحول يتبع للنموذج الذي تمثله المفردة *IDENT* التي حددناها سابقاً.

نظرية: لكل لغة منتظمة أوتومات منته.

نتيجة: كل أوتومات منته يكافئ برنامج حاسوبي.

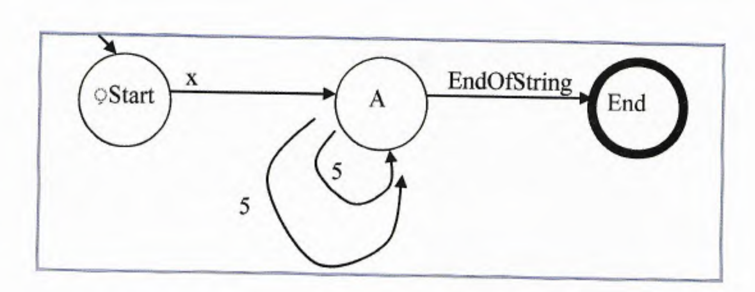
فعلى سبيل المثال، يمكن تمثيل التعبير المنتظم الذي يعرف المفردة *IDENT* على الشكل التالي:



تعريف: يمكننا تعريف أوتومات منته بشكل أولي وسنعود لهذا الموضوع في قسم لاحق) على أنها بيان من مجموعة منتهية من الحالات والوصلات.

- ✓ هناك أوتومات تتميز بوضع لاصقات على الحالات لتعريفها، وهناك أوتومات تتميز بوضع اللاصقات على الناقلات. في كلا الحالتين نحصل على نفس النتيجة.
- ✓ لكل أوتومات حالة ابتدائية.
- ✓ لكل أوتومات مجموعة من الحالات النهائية التي يمثل الوصول إليها انتهاء عمل الأوتومات
- ✓ يتم تعريف اللاصقات بالاعتماد على مجموعة من الرموز ومجموعة من العمليات.
- ✓ يبدأ عمل الأوتومات اعتباراً من الحالة الابتدائية وتتبع، من أجل كل رمز مقروء على مدخلها، وصلة مرتبطة بها وذات لاصقة تتكون من الرمز المقروء (في حال كانت اللاصقات مثبتة على النواقل).
- ✓ في حال تمثيل الأوتومات لتعبير نظامي نقول عن كلمة أنها مقبولة من أوتومات إذا استطعنا الوصول (انطلاقا" من الحالة الابتدائية) إلى حالة نهائية باستخدام رموز وحروف الكلمة.

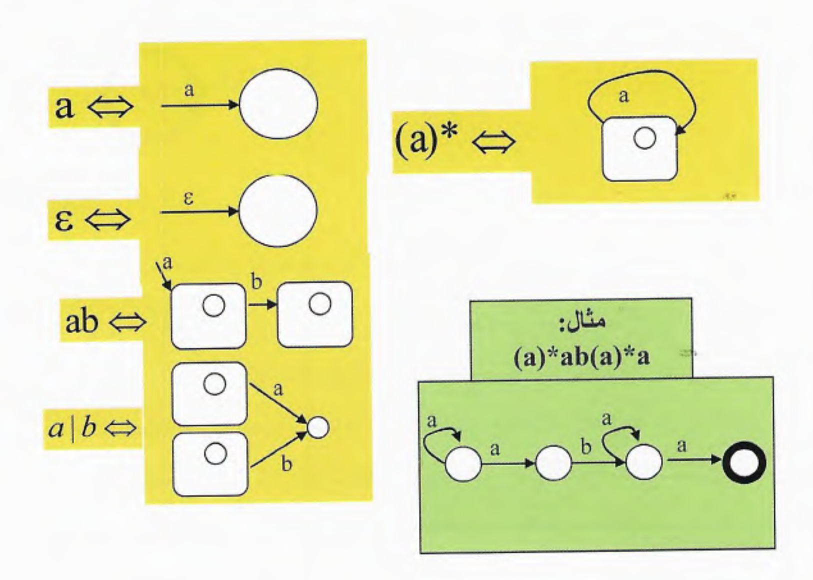
فعلى سبيل المثال يتم التأكد من أن الكلمة *x55* مقبولة من الأوتومات السابقة الموضحة في الشكل بتمريرها حرفاً حرفاً ابتداءاً من الحالة الابتدائية. نلاحظ أنها تصل بعد ثلاث انتقالات إلى الحالة النهائية:



وبما أن كل أوتومات تكافئ برنامج حاسوبي، فيمكننا الآن افتراض وجود إجرائية *GetNext()* تقرأ الحرف التالي من سلسلة الحروف وتضع هذا الحرف في متحول *Symbol*. ولنكتب إجرائية *IsIdentifier()* التي تقوم بالتحقق من كون الكلمة المقروءة هي عبارة عن اسم متحول:

```
int IsIdentifier( )
{
        GetNext( );
        if (Symbol is a Letter)
        {
                GetNext( );
                while ((Symbol is a Letter) | | (Symbol is a Digit)) /*Case Switch*/
                        GetNext( );
                if (Symbol = = ' ')         /* Space*/
                        return TRUE;
                else
                        return Error( );
        }
        else    return Error( );
}
```

لنلق نظرة الآن على قواعد تحويل تعبير منتظم إلى أوتومات والتي يمكن التعبير عنها بخوارزمية تحويل والتي نعرضها هنا بشكل بياني فقط:



بالنتيجة، نستطيع تحويل تعبير منتظم إلى أوتومات منته حتمي، مما يعني أنه سيكون مكافئ لبرنامج حاسوبي وأننا سنستطيع توليد برنامج حاسوبي. وعليه بإمكاننا أتمتة عملية كتابة المحلل من خلال الخطوات التالية:

- ✓ تعريف التعابير المنتظمة المعبرة عن مفردات اللغة التي تمتلك نماذج.
- ✓ استخدام خوارزمية تحويل تعبير منتظم إلى أوتومات.
- ✓ توليد البرنامج الحاسوبي المكافئ للأوتومات وهو البرنامج الذي سيقوم بمعالجة النص البرمجي المصدري وقراءته وتحديد أنواع كلماته.

1 - 3 - 4 - أخطاء المفردات

تكون علاقة المحلل المفرداتي مع الأخطاء قاصرة. إذ يمكن للمحلل المفرداتي التعرف على عدد محدود من الأخطاء:

- ✓ خطأ في كتابة رمز من الرموز التي لا تمتلك نموذج ولا توجد قاعدة لتعريفها، كالرموز البسيطة أو الكلمات المفتاحية.
- ✓ خطأ في تطبيق قاعدة من قواعد تعريف المفردات، ككتابة متحول له الشكل *0.55X*، حيث يظهر وضوحاً أن هذا الرمز ليس كلمة مفتاحية، ولا رمز بسيط، ولا يخضع لا لقاعدة تعريف *IDENT* (فهو ليس متحول)، ولا لقاعدة تعريف *REAL* (فهو ليس عدد حقيقي).

تتم معالجة الأخطاء عموماً اعتماد إما على طريقة تسجيل الحرف الذي سبب الخطأ وإهماله والمتابعة كما سنرى لاحقاً عند التطرق لمعالجة الأخطاء في المحلل القواعدي، حيث سننظر للأمر على نحو أكثر عمومية. أو بمحاولة تصحيح بعض الأخطاء في الرموز البسيطة والكلمات المفتاحية وهو أمر تحاول بعض محررات النصوص المرتبطة بالمترجمات تقديمه، كأن يتم تصحيح *whle* بكتابة *while* في حال أتت هذه الكلمة في بداية الجملة وتلتها جملة تعبر عن حلقة تكرارية. إذ يعتبر محرر النصوص في هذه الحالة أن الكلمة الصحيحة هي الكلمة المفتاحية التي تحتاج لأقل عدد ممكن من التغييرات حتى تتحول من كلمة خطأ إلى كلمة صحيحة في المكان الذي تظهر فيه.

1 - 4 - تمارين

السـؤال 1:

ماهي اللغات التي توصفها التعابير المنتظمة التالية المُعرَّفة على الأبجدية $\{a,b\}$:

- ✓ $a\ (a + b)^*\ b$
- ✓ $aab\ (aa|bb)^{+\ +}$
- ✓ $(aa)^*\ a$

الجواب 1:

- ✓ اللغة التي تبدأ جميع كلماتها بالحرف a وتنتهي بالحرف b.
- ✓ اللغة التي تبدأ جميع كلماتها بالسلسلة aab ومن ثم يليها تكرار واحد على الأقل لثنائية aa أو من bb
- ✓ اللغة التي لا تحتوي كلماتها إلا على حرف a ويكون طول كلماتها مفرداً.

السـؤال 2:

ما هي التعابير المنتظمة التي يمكن أن تعرف اللغات التالية:

- ✓ اللغة على الأبجدية $\{c,b,a\}$ والتي تبدأ جميع كلماتها بالحرف a.
- ✓ الأعداد الصحيحة من مضاعفات 5.

الجواب 2:

- ✓ $a\ (a + b + c)\ *$
- ✓ $(0 + 1 + 2 + 3 + 4 + 5 + 6 + 7 + 8 + 9)*\ (5\ |$

1 - 5 - الأوتومات واللغات الصورية

1 - 5 - 1 - الأوتومات المنتهي

تعريف: نُعرِّف **"الأوتومات المنتهي"** *(Finite Automaton)*، بأنها خماسية $A = (Q, q_0, F, \Sigma, \Delta)$ حيث:

a. Q : مجموعة منتهية من الحالات.

b. q_0 : حالة من Q ندعوها الحالة الابتدائية.

c. F : مجموعة حالات محتواة في Q وندعوها الحالات النهائية.

d. Σ : أبجدية تتألف من مجموعة من الرموز (بما فيها ε الرمز الفارغ) والتي ندعوها رموز الدخل.

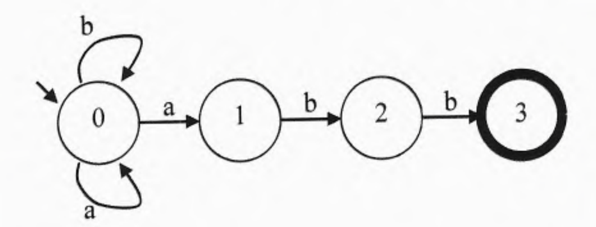
e. Δ : تابع انتقال معرف بالشكل التالي:

$$\Delta : Q \times \Sigma \to 2^Q : \Delta(q_i, a) = \{q_{i1}, \dots, q_{ih}\}$$

مثال: لتكن الأوتومات $A = (\{0,1,2,3\}\ 0, \{4\}\ \{a,b\}\ \Delta)$

حيث يكون التابع: $\begin{cases} \Delta(0,a) = \{0,1\} \\ \Delta(0,b) = 0 \\ \Delta(1,b) = 2 \\ \Delta(2,b) = 3 \end{cases}$

وتكون الأوتومات ممثلة بالشكل التالي:



كما يمكن تمثيل التابع السابق بمصفوفة ندعوها مصفوفة الانتقال وتكون على الشكل التالي:

State / Alphabet	*a*	*b*
0	0,1	0
1	–	2
2	–	3
3	–	–

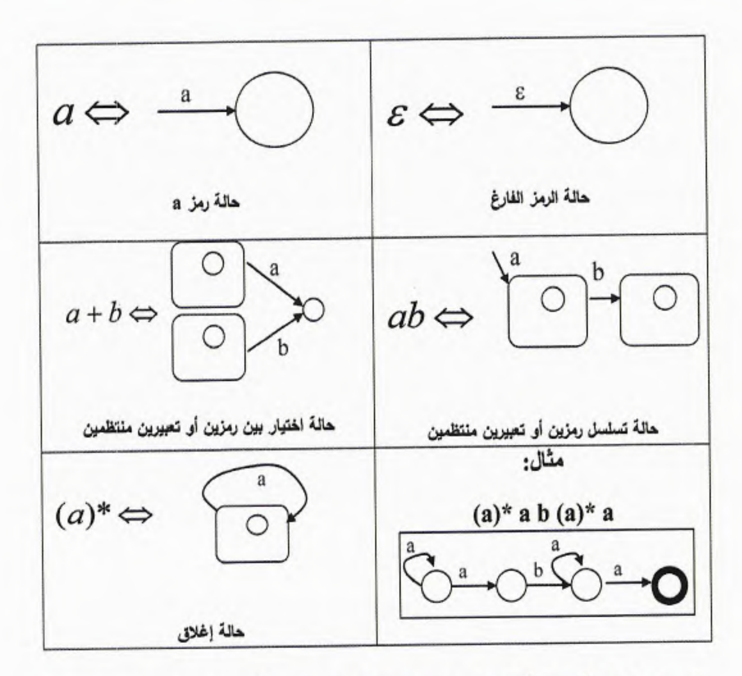
تعريف: نُعرِّف **"اللغة التي تقبلها أوتومات"** بأنها مجموعة السلاسل التي تسمح بالمرور من الحالة الابتدائية إلى إحدى الحالات النهائية للأوتومات.

مثال: في حالة الأوتومات السابقة، تقبل هذه الأوتومات اللغة الممثلة بالتعبير المنتظم: $(a + b)^* abb$

بشكل عام، هناك تقابل بين الأوتومات المنتهي والخوارزمية، فكل أوتومات منته يعبر عن خوارزمية أو عن برنامج منته. وتزداد سهولة التعبير عن أوتومات بخوارزمية، عندما تكون الأوتومات من النوع المنتهي والحتمي وهو ما سنعرّفه لاحقاً.

1 - 5 - 2 - تحويل تعبير منتظم إلى أوتومات منته لاحتمي

لبناء أوتومات منتهي اعتباراً من تعبير منتظم يمكننا اعتماد الخوارزمية الموضحة في الشكل التالي والتي تحدد الأوتومات المقابل لكل نوع من أنواع التعابير المنتظمة:

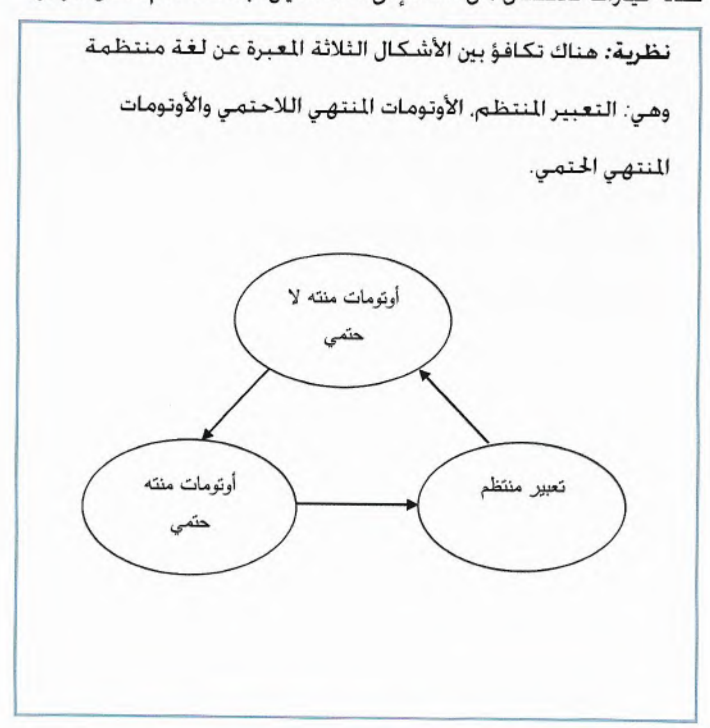


1 - 5 - 3 - تحويل أوتومات منته لاحتمي إلى أوتومات منته حتمي

تعريف: نقول عن "الأوتومات المنتهي" أنه "**حتمي**" *(Deterministic)* في حال لم يكن في أبجديته الرمز ε وكان تابع الانتقال معرفاً بالشكل: $\Delta : Q \times \Sigma \rightarrow Q : \Delta(q_i, a) = q_j$ بحيث يتم الانتقال بأي رمز من حالة إلى حالة واحدة فقط.

يسمح الانتقال من أوتومات منته لاحتمي إلى أوتومات منته حتمي بتجنب حالات الرجوع إلى الخلف عند التأكد من قبول الأوتومات لكلمة من اللغة المعرفة بهذه الأوتومات. يعود السبب في ذلك إلى عدم وجود عدة خيارات للانتقال من حالة إلى حالة تليها باستخدام نفس الرمز.

نظرية: هناك تكافؤ بين الأشكال الثلاثة المعبرة عن لغة منتظمة وهي: التعبير المنتظم، الأوتومات المنتهي اللاحتمي والأوتومات المنتهي الحتمي.



يجري تحويل أوتومات منته لاحتمي إلى أوتومات منته حتمي اعتماداً على الخوارزمية التالية:

Input: *Non Deterministic Finite Automaton*

$A1 = (Q1, q_0, F1, \Sigma, \Delta 1)$

Output: *Deterministic Finite Automaton*

$A2 = (Q2, q_0, F2, \Sigma, \Delta 2)$

For each $p \in Q1$ *and for each* $a \in \Sigma$ ***Do***

Add to transition table all the composite states produced by $\Delta 1(p, a)$

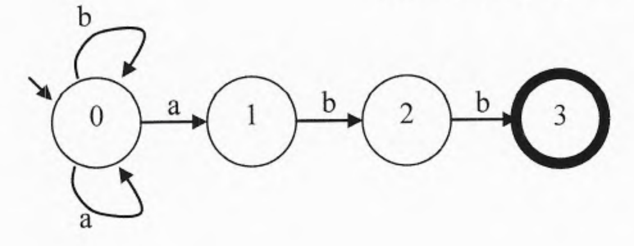
All states with at least one final state, become final states

Renumber the states

مثال: لنأخذ حالة $A = (\{0,1,2,3\}\ 0, \{4\}\ \{a,b\}\ \Delta)$

حيث التابع: $\begin{Bmatrix} \Delta(0,a) = \{0,1\} \\ \Delta(0,b) = 0 \\ \Delta(1,b) = 2 \\ \Delta(2,b) = 3 \end{Bmatrix}$ حيث يكون للأوتومات

الشكل الموضح فيما يلي:



State / Alphabet	a	b
0	0,1	0
1	–	2
2	–	3
3	–	–

➔

State / Alphabet	a	b
0	0,1	0
1	–	2
2	–	3
3	–	–
0,1	0,1	0,2

➔

State / Alphabet	a	b
0	0,1	0
1	–	2
2	–	3
3	–	–
0,1	0,1	0,2
0,2	0,1	0,3

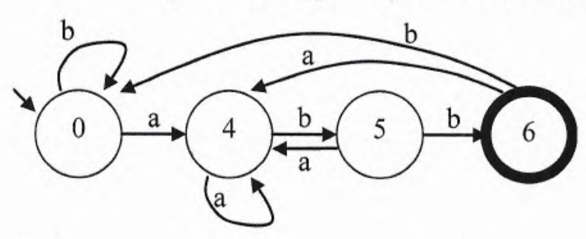
➔

State / Alphabet	a	b
0	0,1	0
1	–	2
2	–	3
3	–	–
0,1	0,1	0,2
0,2	0,1	0,3
0,3	0,1	0

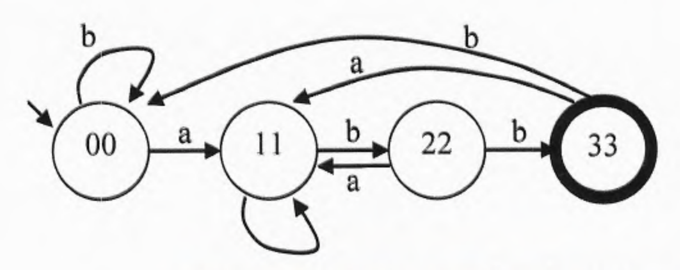
➔

State / Alphabet	a	b
0	4	0
1	–	2
2	–	3
3	–	–
4	4	5
5	4	6
6	4	0

ويكون الأوتومات المنتهي الحتمي الناتج كما يلي:



أما جزء الأوتومات المؤلف من الحالات *1* و*2* و*3* فيمكن إهماله لأننا لا يمكن أن نصل إليه اعتباراً من الحالة الابتدائية. لذا يصبح هذا الجزء إضافي ولا معنى له ويمكن إهماله. وتكون الأوتومات الناتجة والمرسومة في الشكل السابق هي الأوتومات المنتهي الحتمي الناتج والتي يمكن إعادة ترقيم حالاتها كما يلي:



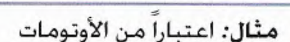
1 - 5 - 4 - تحويل أوتومات منته إلى تعبير منتظم

لنفرض أن لدينا لغة L تقبلها الأوتومات $A = (Q, q_0, F, \Sigma, \Delta)$

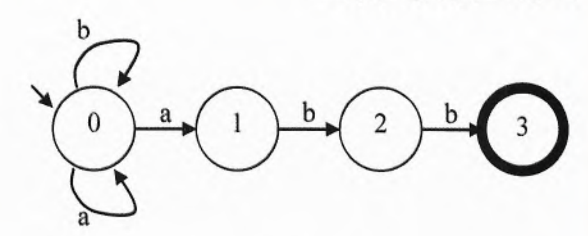
ولنفرض أن L_i هي اللغة التي يمكن أن تقبلها الأوتومات A إذا اعتبرنا أن الحالة الابتدائية لجميع سلاسلها هي q_i. يمكننا عندها كتابة جملة معادلات التي تربط جميع الأجزاء L_i التي تكون اللغة الأصلية L التي تقبلها A، كما يلي:

- كل انتقال من الشكل $\Delta(q_i, a) = q_j$ يسمح بكتابة المعادلة:
  $L_i = aL_j$.
- من أجل كل $q_i \in F$ لدينا المعادلة $L_i = \varepsilon$.
- يمكن تجميع جميع المعادلات $L_i = \alpha$ و $L_i = \beta$ بالشكل
  $L_i = \alpha \mid \beta$.
- يتم حل جملة المعادلات الناتجة عن طريق التعويض لحساب L_0.

خاصة مهمة: $\left[L = uL \mid v \Rightarrow L = u^{*}v\right]$



مثال: اعتباراً من الأوتومات



يمكننا أن نستنتج جملة المعادلات التالية:

$$\begin{cases} L_0 = aL_0 \mid bL_0 \mid aL_1 \\ L_1 = bL_2 \\ L_2 = bL_3 \end{cases} \rightarrow \begin{cases} L_0 = aL_0 \mid bL_0 \mid aL_1 \\ L_1 = bbL_3 \end{cases}$$

$$\rightarrow \{L_0 = aL_0 \mid bL_0 \mid abbL_3 \rightarrow (a \mid b)^{*} abb$$

1 - 5 - 5 - الأوتومات ذات المكدس

هناك بعض اللغات التي لا يمكن التعرف عليها باستخدام أوتومات منته ولا يمكن توصيفها باستخدام تعبير منتظم. تكون هذه اللغات أوسع من اللغات المنتظمة وندعوها باللغات خارج السياق. من الأمثلة النمطية على لغة من هذا النوع، اللغة المبنية على الرمزين a وb ولها الشكل $a^n b^n$ من أجل n صحيح موجب والتي تنتمي إليها سلاسل مثل ab أو $aaabbb$ ... الخ. ويبدو واضحاً من شكل اللغة $a^n b^n$ أن

محاولة تمثيلها بأوتومات ستصطدم بعائق الحاجة إلى وجود ذاكرة تسمح بتخزين عدد المرات (وهي n) التي ظهر فيها الرمز a في الكلمة للتأكد من أن b ستظهر بنفس العدد من المرات. وهو أمر غير ممكن في أي أوتومات منته كون n غير ثابتة وغير محددة ويمكن أن تسعى إلى اللانهاية. لتمثيل مثل هذه اللغات، نستخدم أداة ذات إمكانيات إضافية وهي الأوتومات ذات المكدس.

تعريف: نعرّف **"الأوتومات ذات المكدس"** *(Push Down Automaton)*، عبارة عن سباعية $P = (Q, q_0, F, \Sigma, \Gamma, \tau, \Delta)$

حيث:

f. Q: مجموعة منتهية من الحالات.

g. q_0: حالة من Q ندعوها الحالة الابتدائية.

h. F: مجموعة حالات محتواة في Q وندعوها الحالات النهائية.

i. Σ: الأبجدية وتتألف من مجموعة من الرموز.

j. Γ: أبجدية المكدس.

k. τ: رمز قعر المكدس.

l. Δ: تابع انتقال معرف بالشكل التالي:

$$\Delta : Q \times \Sigma^* \times \Gamma^* \rightarrow Q \times \Gamma^*$$

تعريف: تكون اللغة مقبولة من أوتومات ذات مكدس إذا كانت مجموعة السلاسل التي تنتمي لهذه اللغة تنتقل عند بدايتها من حالتي: مكدس فارغ يحوي رمز قعر المكدس وحالة الأوتومات الابتدائية، لتصل عند نهايتها (وبعد قراءة كافة رموزها) إلى إحدى حالتين:

- حالة نهائية من حالات الأوتومات.
- أو عودة إلى حالة مكدس فارغ.

مثال: لنعرّف الأوتومات ذات المكدس التي تمثل $a^n b^n$ من أجل n صحيح موجب. تكون الأوتومات مؤلفة من العناصر التالية:

$$\begin{cases} Q = \{s,p,q\} \\ q_0 = s \\ F = \{q\} \\ \Sigma = \{a,b\} \\ \Gamma = \{A,\$\} \\ \tau = \$ \end{cases} , \quad \Delta \begin{cases} (s,a,\varepsilon) \to (s,A) \\ (s,b,A) \to (p,\varepsilon) \\ (p,b,A) \to (p,\varepsilon) \\ (s,\varepsilon,\$) \to (q,\varepsilon) \\ (p,\varepsilon,\$) \to (p,\varepsilon) \end{cases}$$

لتكن لدينا الكلمة: *aaabbb*، تكون حركة الأوتومات الناجمة عن قراءة الكلمة وقبولها ممثلة على النحو التالي:

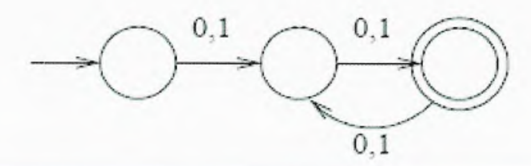
Stack	Input	State	Transition
\$	aaabbb	s	$(s,a,\varepsilon) \to (s,A)$
\$A	aabbb	s	$(s,a,\varepsilon) \to (s,A)$
\$AA	abbb	s	$(s,a,\varepsilon) \to (s,A)$
\$AAA	bbb	s	$(s,b,A) \to (p,\varepsilon)$
\$AA	bb	p	$(p,b,A) \to (p,\varepsilon)$
\$A	b	p	$(p,b,A) \to (p,\varepsilon)$
\$		p	$(p,\varepsilon,\$) \to (p,\varepsilon)$
\$		q	ACCEPTED

1 - 6 - تمارين

السؤال الأول:

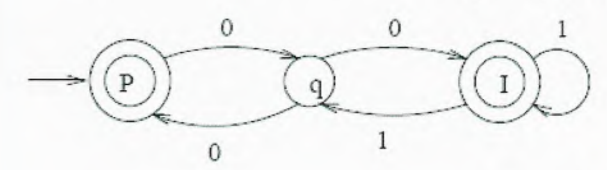
أعط الأوتومات المنتهي الحتمي التي تقبل سلاسل لها طول زوجي *(عدا السلسلة الفارغة)*.

الجواب الأول:

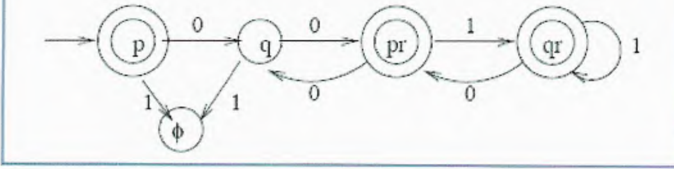


السؤال الثاني:

أعط الأوتومات المنتهي الحتمي المكافئ للأوتومات المنتهي اللاحتمي التالي:



الجواب الثاني:

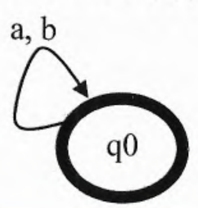


السؤال الثالث:

هل اللغة a^*b^* هي لغة منتظمة؟

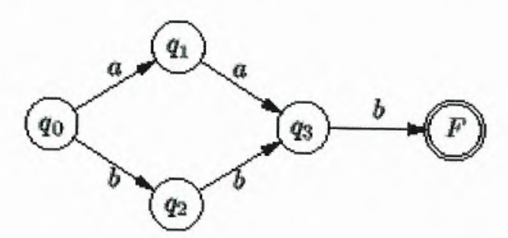
الجواب الثالث:

نعم لأن بإمكاننا رسم أوتومات منته حتمي لها وهو



السؤال الرابع:

أعط التعبير المنتظم المعبر عن الأوتومات المنتهي الحتمي التالي:



الجواب الرابع:

(aa+bb)b

1 - 7 - التحليل القواعدي *(Syntax Analysis)*

تمتلك كل لغة من لغات البرمجة مجموعة من القواعد التي تحدد شكل وبنية الجمل الصحيحة التي يمكن كتابتها باستخدام هذه اللغة. فعلى سبيل المثال، يكون برنامج مكتوب بلغة *C* مؤلفاً من مقاطع *(Blocs)* كل مقطع منها مؤلف من مجموعة تعليمات *(Instructions)* وهكذا دواليك، وذلك وفقاً لقواعد ندعوها القواعد الصرفية *(Syntax Rules)*، وبحيث تشكل مجموعة القواعد الصرفية، نحواً صرفياً *(Grammar)*.

يتلقى المحلل القواعدي عادةً سلسلةً من الكلمات التي قام المحلل المفرداتي بتمييزها. يعمل المحلل القواعدي على التحقق من أن السلسلة (الجملة) التي تشكلت لديه متوافقة مع النحو الصرفي الخاص بلغة البرمجة التي يُفترض أن النص مكتوب بها. تصبح المسألة على النحو التالي:

- بفرض أن لدينا نحواً صرفياً *G*.
- بفرض أن لدينا سلسلة *m* من المفردات.

هل تنتمي *m* إلى اللغة التي يولدها *G*؟

يعتمد التحليل الصرفي الذي نتناوله في هذا الفصل على بناء شجرة الاشتقاق *(Derivation Tree)*. إلا أن بناء هذه الشجرة يتم عادةً وفق طريقتين للمسح والتحليل: تحليل نازل *(Top-Down Parsing)* وتحليل صاعد *(Bottom-Up Parsing)*.

1 - 7 - 1 - النحو الصرفي ومفهوم شجرة الاشتقاق

سنطرح السؤال التالي: في حال كان لدينا لغة ما، كيف يمكن توصيف جميع السلاسل المقبولة التي تنتمي إليها؟

لقد سبق وطرحنا هذا السؤال وأجبنا عليه في فصول سابقة عندما تعاملنا مع التعابير المنتظمة واعتبرناها وسيلة للتعبير. إلا أننا وجدنا أيضاً أن التعابير المنتظمة تصلح للتعبير عن لغات منتظمة، وأن هذه اللغات لا تشكل إلا طيفاً ضيقاً جداً من اللغات التي يمكن أن نتعامل معها. ووجدنا أيضاً أن هناك لغات أكثر اتساعاً (كاللغة $a^n b^n$) لا يمكن التعبير عنها بتعبير منتظم. لذا نحتاج في هذه الحالة إلى أداة أكثر قوة للتعبير عن مثل هذه اللغات. تتمثل هذه الأداة بما ندعوه النحو الصرفي.

a. النحو الصرفي

تمتلك الجملة في أي لغة، بنية محددة بقواعد اللغة، كما هو الحال في اللغة العربية حيث يمكننا القول (مع التبسيط) أن الجملة الفعلية تتألف من فعل يليه فاعل. ويمكن أن نكتب ما سبق على الشكل التالي:

جملة فعلية = فعل فاعل

ندعو ما سبق قاعدة صرفية وتكون هذه القاعدة عادةً، جزءاً من مجموعة القواعد التي تضبط اللغة والتي ندعوها النحو الصرفي.

لنفترض الآن أن بإمكان الفعل والفاعل أن يأخذا عدة قيم كما هو الحال فيما يلي:

فعل= أكل | شرب
فاعل= خالد | عمر

تمكننا هذه القيم من تشكيل أربع جمل عربية صحيحة من اللغة:

أكل خالد | شرب خالد | أكل عمر | شرب عمر

بالتالي، نستطيع اعتباراً من القواعد الصرفية للغة، اشتقاق الجمل التي تمثلها هذه القواعد.

عموماً، تحتوي أية لغة على عدد لانهائي من الجمل ولكنها لا تحتاج إلى عدد لانهائي من القواعد. فعلى سبيل المثال، يمكننا لتوليد أي عدد صحيح باستخدام القاعدة التالية:

Number= Digit | Digit Number
Digit = 0|1|2|3|4|5|6|7|8|9

وعليه، يمكننا تعريف النحو الصرفي والقواعد الصرفية التي تنتمي إليه على النحو التالي:

تعريف: نعرف النحو الصرفي *(Grammar)* بأنه رباعية $G=(V_T,V_N,S,P)$ حيث:

V_T: مجموعة غير فارغة من رموز الأبجدية *(Terminals)* التي ندعوها الرموز الأولية.

V_N: مجموعة غير فارغة من الرموز الوسيطة *(Non Terminals)*

حيث $V_T \cap V_N = \phi$.

S: رمز البداية ويكون $S \in V_N$.

P: مجموعة القواعد الصرفية الخاصة بالنحو G.

تعريف: تحدد القاعدة الصرفية $\alpha \rightarrow \beta$ أنه يمكننا استبدال سلسلة الرموز α (حيث $\alpha \in (V_T \cup V_N)^+$) بسلسلة الرموز β (حيث $\beta \in (V_T \cup V_N)^+$). ندعو α الجزء اليساري من القاعدة، في حين ندعو β الجزء اليميني منها.

لنأخذ مثلاً النحو الصرفي التالي:

$$G\begin{cases} V_T = \{a, b\} \\ V_N = \{A\} \\ S : A \\ P : A \rightarrow aAb \mid \varepsilon \end{cases}$$

يمثل هذا النحو الصرفي اللغة التي سبق وتطرقنا إليها وهي ($a^n b^n$) وهو ما سنستعرضه في الفقرة اللاحقة.

b. شجرة الاشتقاق

تعريف: ندعو «**اشتقاقاً**» *(Derivation)* التطبيق المتتالي لمجموعة من القواعد اعتباراً من كلمة تنتمي إلى $(V_T \cap V_N)^+$

نستخدم إشارة السهم ($\rightarrow$) للإشارة إلى اشتقاق ناجم عن تطبيق قاعدة واحدة. كما نستخدم إشارة السهم ($\overset{*}{\rightarrow}$) المزود بنجمة للإشارة إلى اشتقاق ناتج عن تطبيق عدد *n* من القواعد حيث $n \geq 0$.

أمثلة:

$$G\begin{cases} V_T = \{a,b\} \\ V_N = \{A\} \\ S : A \\ P : A \to aAb \mid \varepsilon \end{cases} \Rightarrow S \to aAb \to aaAbb \to aaaAbbb \to aaabbb \Rightarrow S \overset{*}{\to} aaabbb$$

$$\begin{array}{l} \text{Number} = \text{Digit} \mid \text{Digit Number} \\ \text{Digit} = 0|1|2|3|4|5|6|7|8|9 \end{array} \Rightarrow \text{Number} \to 5\ \text{Number} \to 52 \Rightarrow \text{Number} \overset{*}{\to} 52$$

تعريف: من أجل نحو صرفي $G = (V_T, V_N, S, P)$، نعرف *L(G)* اللغة التي يولدها هذا النحو، كما يلي:

$$L(G) = \{w \in (V_T)^* : S \overset{*}{\to} w\}$$

تعريف: من أجل نحو صرفي $G = (V_T, V_N, S, P)$، ندعو شجرة الاشتقاق الشجرة التي يكون:

a. جذرها هو الرمز *S*.

b. أوراقها هي مجموعة الرموز الأولية المنتمية إلى V_T أو الرمز ε .

c. عقدها هي مجموعة الرموز الوسيطة المنتمية إلى V_N .

d. تكون العقد التالية *(العقد الأبناء)* لعقدة α هي مجموعة العقد $\beta_1,...,\beta_n$ إذا وفقط إذا كان: $\alpha \to \beta_1,...,\beta_n$



يكون **الاشتقاق يسارياً** *(Leftmost Derivation)* إذا استبدلنا عند تنفيذ كل عملية اشتقاق للجملة اليمينية، الرمز الوسيط *(non terminal symbol)* الموجود على أقصى يسار هذه الجملة بالقاعدة المناسبة له.

يكون **الاشتقاق يمينياً** *(Rightmost Derivation)* إذا استبدلنا، عند تنفيذ كل عملية اشتقاق للجملة اليمينية، الرمز الوسيط *(non terminal symbol)* الموجود على أقصى يمين هذه الجملة بالقاعدة المناسبة له.

مثال: ليكن لدينا النحو الصرفي التالي:

$$G = (\{a, b, c\}\ \{S, T\}\ S, P)\ P : \begin{cases} S \rightarrow aTb \mid c \\ T \rightarrow cSS \mid S \end{cases}$$

تكون شجرة الاشتقاق *(الناجمة عن الاشتقاق اليميني أو اليساري)* للكلمة *(accacbb)* هي الشجرة التالية:

$$\text{Leftmost} : S \rightarrow aTb \rightarrow acSSb \rightarrow accSb \rightarrow accaTbb \rightarrow accaSbb \rightarrow accacbb$$
$$\text{Rightmost} : S \rightarrow aTb \rightarrow acSSb \rightarrow acSaTbb \rightarrow acSaSbb \rightarrow acSacbb \rightarrow accacbb$$

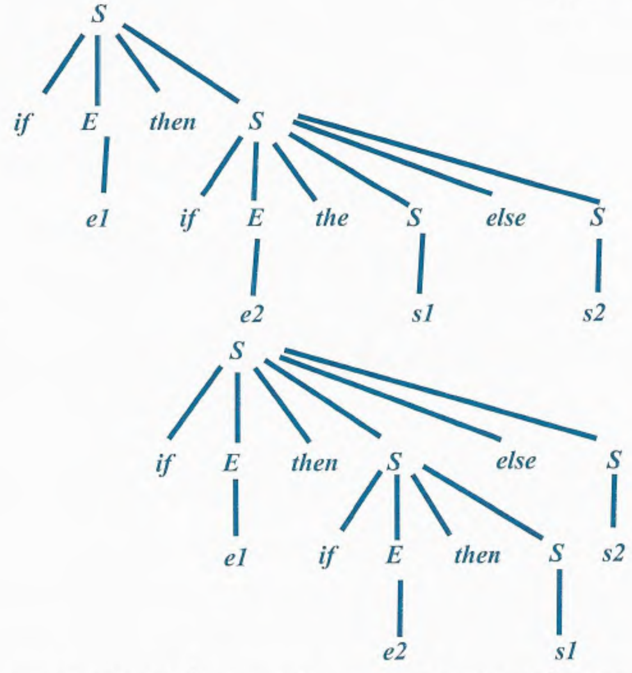
قد تتسبب القواعد الصرفية الموضوعة بتوليد أكثر من شجرة اشتقاق يميني أو أكثر من شجرة اشتقاق يساري ندعو هذه المشكلة بمشكلة الغموض *(Ambiguity)*. فعلى سبيل المثال، لنأخذ قواعد الجملة الشرطية التالية:

$$S \rightarrow \textit{if } E \textit{ then } S$$
$$\mid \textit{if } E \textit{ then } S \textit{ else } S$$

في حال قمنا باشتقاق الجملة:

if e1 then s1 if e2 then s2 else s3

فسنجد أننا، في حالة الاشتقاق اليساري، سنحصل على شجرتين مختلفتين:



ما يعني أن القواعد تعاني من حالة غموض وأن اشتقاقها اليساري (أو اليميني) سيؤدي إلى الوصول إلى حالة يصعب فيها التنبؤ بكيفية إتمام عملية الاشتقاق. ومن الواضح هنا هو أن الغموض ناجم عن عدم تحديد القواعد التي تضبط لتبعية تعليمة *else* إلى تعليمة *if* في حال وجود عدة بنى لتعليمة *if* متداخلة ببعضها البعض.

تنفيذ التحليل القواعدي

يستقبل المحلل القواعدي سلسلة من المفردات (رموز نهائية) من المحلل المفرداتي. وتكون مهمته تحديد صحة هذه الجملة وفقاً للقواعد الصرفية التي يعمل بها. لذا يعمل المحلل القواعدي على بناء شجرة اشتقاق جملة اعتباراً من تلك القواعد الصرفية، ويعمل على التأكد من عدم وجود أي حالات غموض فيها.

يعتمد المحلل الصرفي في تحليله للجملة على إحدى مقاربتين: تحليل نازل يبدأ من رمز البداية S للقواعد ويسعى بتطبيقه للاشتقاقات الوصول إلى الجملة المطلوب التحقق منها. وآخر صاعد يبدأ من الجملة ويسعى لتعويض سلاسل الكلمات المتشكلة من الرموز النهائية، بالرموز الوسيطة التي تعبر عنها وفقاً للقواعد، تمهيداً للوصول إلى رمز البدية S.

تحليل نازل من الأعلى إلى الأسفل

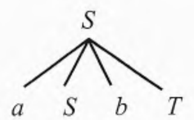
المبدأ: بناء شجرة اشتقاق الجملة التي نقوم بالتأكد من صحتها، بشكل نازل يبدأ من رمز البداية S للقواعد ويسعى من خلال تطبيقه للاشتقاقات، إلى الوصول للجملة المطلوبة.

أمثلة

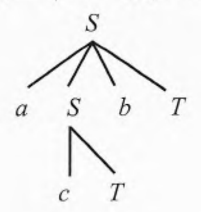
المثال الأول: لتكن لدينا القواعد التالية، حيث يعبر S عن رمز البداية، وحيث T رمز وسيط:

$$\boxed{\begin{array}{l} S \rightarrow aSbT \mid cT \mid d \\ T \rightarrow aT \mid bS \mid c \end{array}}$$

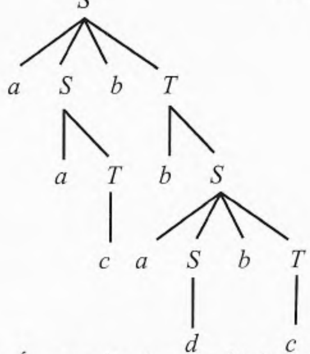
نريد تحليل الكلمة: $w=accbbadbc$. لنقلع من جذر الشجرة وهو الرمز S. تؤدي قراءة الكلمة a إلى التقدم في عملية بناء الشجرة لنحصل على: $W=\underline{\mathbf{a}}ccbbadba$



تؤدي قراءة الكلمة الثانية c إلى التقدم خطوة أخرى: $W=\underline{\mathbf{ac}}cbbadba$



وهكذا حتى نصل إلى:



طالما أننا وجدنا شجرة اشتقاق، فهذا يعني أن الجملة صحيحة قواعدياً وفق القواعد المحددة في بداية المثال.

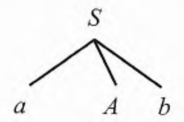
المثال الثاني: لتكن لدينا القواعد التالية، حيث يعبر S عن رمز البداية، وحيث A رمز وسيط:

$$\boxed{\begin{array}{l} S \rightarrow aAb \\ A \rightarrow d \ \ | c \end{array}}$$

نريد تحليل الكلمة: $w=acb$

لنقلع من جذر الشجرة وهو الرمز S، فتؤدي قراءة الكلمة a إلى التقدم في عملية بناء الشجرة لنحصل على:

$W=\underline{\mathbf{a}}cb$



ولكن عند قراءة c لن نستطيع تحديد فيما إذا كان علينا أخذ القاعدة $(A\grave{a}c\ d)$ أو القاعدة $(A\grave{a}c)$. لتحديد الأمر، يتوجب علينا قراءة الكلمة التالية، وهي (b)، وإعطاء إمكانية الرجوع إلى الخلف بحيث نجرب القاعدة الأولى $(A\grave{a}c\ d)$ ونتابع. وفي حال لم يجر الأمر بالشكل الصحيح نعود إلى الخلف، ونعيد بناء هذا الجزء مع القاعدة الثانية. تؤدي العودة إلى الخلف *(backtracking)* إلى ارتفاع كلفة عملية التحليل.

المثال الثالث: لتكن لدينا القواعد التالية، حيث يعبر S عن رمز البداية:

$$\boxed{S \rightarrow aSb \mid aSc \mid d}$$

نريد تحليل الكلمة: $w=aaaaaaadbbcbbbc$



من الواضح هنا أننا نحتاج منذ البداية لتحديد القاعدة التي يجب الانطلاق منها. للأسف لن نستطيع تحديدها إلا في نهاية الاشتقاق مما سيجعلنا نبني ثم نعود إلى الوراء عدة مرات قبل اكتشاف الشجرة الصحيحة.

المثال الرابع: لتكن لدينا القواعد التالية التي توصف تعابير حسابية، حيث يعبر E عن رمز البداية، وحيث F وT رموز وسيط:

$$\begin{array}{|l|}\hline E \rightarrow T\,E' \\ E' \rightarrow +\,T\,E' \mid \varepsilon \\ T \rightarrow F\,T' \\ T' \rightarrow *\,F\,T' \mid \varepsilon \\ F \rightarrow (E) \mid Id \\ \hline \end{array}$$

من الواضح هنا أننا لا نستطيع المتابعة بالأسلوب التجريبي السابق، وبأننا بحاجة لمنهجية تساعدنا في تنفيذ عملية التحليل.

يمكننا أن نلخص منهجية التحليل النازل التي نحتاجها بما يلي:

نحتاج إلى جدول يحدد لنا، في حال قمنا بقراءة كلمة ما، وفي حال كنا في مرحلة اشتقاق معينة عند عقدة معينة للشجرة ممثلة برمز وسيط، ما هي القاعدة الخاصة بهذا الرمز التي يجب اختيارها لمتابعة بناء الشجرة.

a. التحليل *LL(1)*

في بداية هذا التحليل، نحتاج لحساب مجموعتين: مجموعة الرموز الأولى *(First)*، ومجموعة الرموز اللاحقة *(Follow)*. ومن ثم سنعتمد على هاتين المجموعتين في بناء الجدول المطلوب.

مجموعة الرموز الأولى *(First)*:

من أجل كل سلسلة α مؤلفة من رموز أولية ورموز وسيطة، نبحث عن $\mathrm{First}(\alpha)$ وهي مجموعة الرموز الأولية التي تبدأ بها السلسلة المشتقة من α. بمعنى آخر نبحث عن الرموز الأولية a بحيث:

$\alpha \rightarrow a\beta^*$ حيث $a\beta^*$ سلسلة مؤلفة من رموز أولية ووسيطة.

مثال:

$$\begin{cases} S \rightarrow Ba \\ B \rightarrow cP \mid bP \mid P \mid \varepsilon \\ P \rightarrow dS \end{cases} \Rightarrow \begin{cases} S \overset{*}{\rightarrow} a \Rightarrow a \in \mathrm{First}(S) \\ S \overset{*}{\rightarrow} cPa \Rightarrow c \in \mathrm{First}(S) \\ S \overset{*}{\rightarrow} bPa \Rightarrow b \in \mathrm{First}(S) \\ S \overset{*}{\rightarrow} dSa \Rightarrow a \in \mathrm{First}(S) \end{cases} \Rightarrow$$

$$\mathrm{First}(S) = \{a, b, c, d\}$$

وتكون خوارزمية بناء المجموعة *First* على النحو التالي:

Repeat

if *(X is non terminal and* $X \rightarrow Y_1 Y_2 ... Y_n$ *is a production where* Y_i *is either terminal or non terminal)* ***then***

{

Add elements of (First(Y_1*)-* $\{\varepsilon\}$ *) to First(X);*

if *(*$\exists j \in [2.\, n]$ *where for each* $i \in [1.\, j-1]$ *we have* $\varepsilon \in \text{First}(Y_i)$*)* ***then***

$$\text{First}(X) = \text{First}(X) \cup (\text{First}(Y_j) - \{\varepsilon\}$$

for *i=1..n* ***do*** ***if*** $\varepsilon \in \text{First}(Y_i)$ ***then*** $\text{First}(X) = \text{First}(X) \cup \{\varepsilon\}$

}

if *(X is non terminal and we have the production* $X \rightarrow \varepsilon$*)* ***then***

$$\text{First}(X) = \text{First}(X) \cup \{\varepsilon\}$$

if *(X is a terminal)* ***then***

$$\text{First}(X) = \{X\}$$

until *First(X) has no changes*

أمثلة:

$$\begin{array}{|l|}\hline E \to T\,E' \\ E' \to +T\,E' \mid \varepsilon \\ T \to F\,T' \\ T' \to *F\,T' \mid \varepsilon \\ F \to (E) \mid Id \\ \hline \end{array} \Rightarrow \begin{array}{|l|}\hline First(E) = First(T) = First(F) = \{(, Id\}) \\ First(E') = \{+, \varepsilon\} \\ First(T') = \{*, \varepsilon\} \\ \hline \end{array}$$

$$\begin{array}{|l|}\hline S \to ABC \\ A \to aA \mid \varepsilon \\ B \to bB \mid cB \mid \varepsilon \\ C \to de \mid da \mid dA \\ \hline \end{array} \Rightarrow \begin{array}{|l|}\hline First(S) = \{a, b, c, d\} \\ First(A) = \{a, \varepsilon\} \\ First(B) = \{b, c, \varepsilon\} \\ First(C) = \{d\} \\ \hline \end{array}$$

مجموعة الرموز اللاحقة *(Follow)*

من أجل كل رمز وسيط *A*، نبحث عن مجموعة الرموز اللاحقة *(Follow(A*). وهي مجموعة الرموز الأولية التي يمكن أن تظهر مباشرةً على يمين الرمز *A* في أي اشتقاق $S \overset{*}{\to} \alpha A a \beta$.

مثال:

وتكون خوارزمية بناء المجموعة *Follow* على النحو التالي:

$$\boxed{\begin{cases} S \to Sc \mid Ba \\ B \to Pa \mid bPb \mid P \mid \varepsilon \\ P \to dS \end{cases}} \Rightarrow \boxed{\begin{cases} S \overset{*}{\to} Sc \Rightarrow c \in Follow(S) \\ S \overset{*}{\to} dSa \Rightarrow a \in Follow(S) \\ S \overset{*}{\to} bdSba \Rightarrow b \in Follow(S) \\ \dots \end{cases}}$$

$$\Rightarrow \boxed{a, b, c \in Follow(S)}$$

Repeat

$\text{Follow}(S) = \text{Follow}(S) \cup \{\$\}$ *(where $ is the end symbol, S is the starting Symbol)*

if *(* $\exists A \rightarrow \alpha B \beta$ *where B is non terminal)* ***then***

$$\text{Follow}(B) = \text{Follow}(B) \cup (\text{First}(\beta) - \{\varepsilon\})$$

if *(* $\exists A \rightarrow \alpha B$ *where B is non terminal)* ***then***

$$\text{Follow}(B) = \text{Follow}(B) \cup \text{Follow}(A)$$

if *(* $\exists A \rightarrow \alpha B \beta$ *where B is non terminal and* $\varepsilon \in \text{First}(\beta)$ *)* ***then***

$$\text{Follow}(B) = \text{Follow}(B) \cup \text{Follow}(A)$$

Until *the sets Follow has no changes*

أمثلة:

$$\begin{array}{l} E \rightarrow T\,E' \\ E' \rightarrow +\,T\,E' \mid \varepsilon \\ T \rightarrow F\,T' \\ T' \rightarrow *\,F\,T' \mid \varepsilon \\ F \rightarrow (E) \mid Id \end{array} \Rightarrow \begin{array}{l} \text{Follow}(E) = \text{Follow}(E') = \{\$\ )\} \\ \text{Follow}(T) = \text{Follow}(T') = \{+, \$\ )\} \\ \text{Follow}(F) = \{*, \$\ )\} \end{array}$$

$$\begin{array}{l} S \rightarrow aSb \mid d \mid SAe \\ A \rightarrow aAdB \mid \varepsilon \\ B \rightarrow bb \end{array} \Rightarrow \begin{array}{l} \text{Follow}(S) = \{b\ a\ e\} \\ \text{Follow}(A) = \{e, d\} \\ \text{Follow}(B) = \{e, d\} \end{array}$$

بناء الجدول

لبناء جدول القواعد M الذي سنستخدمه في عملية التحليل، نعتمد على الخوارزمية التالية:

for *each production* $X \rightarrow Y$ ***do***

{

  for *each* $(a \neq \varepsilon)$ *in First(X)* ***do*** *M[X,a]=* $X \rightarrow Y$

  if *(X*$\varepsilon \rightarrow$ *)* ***then for*** *each b in Follow(X)* ***do*** *M[X,b]=*
$X\varepsilon \rightarrow$

}

مثال:

$$
\begin{array}{|l|}
\hline
E \rightarrow T\,E' \\
E' \rightarrow +\,T\,E' \mid \varepsilon \\
T \rightarrow F\,T' \\
T' \rightarrow *\,F\,T' \mid \varepsilon \\
F \rightarrow (E) \mid Id \\
\hline
\end{array}
\Rightarrow
$$

	Id	+	*	(	)	$
E	$E \rightarrow TE'$			$E \rightarrow TE'$		
E'		$E' \rightarrow +TE'$			$E' \rightarrow \varepsilon$	$E' \rightarrow \varepsilon$
T	$T \rightarrow FT'$			$T \rightarrow FT'$		
T'		$T' \rightarrow \varepsilon$	$T' \rightarrow *FT'$		$T' \rightarrow \varepsilon$	$T' \rightarrow \varepsilon$
F	$F \rightarrow Id$			$F \rightarrow (E)$		



تنفيذ التحليل

لنفرض أن لدينا جملة نريد تحليلها بأسلوب التحليل النازل. ستتعامل خوارزمية التحليل مع ثلاثة مكونات:

a. الشريط *Word* الذي سيحوي الجملة (التي تنتهي بالحرف $) وبحيث يكون للشريط مؤشر *wp* على الحرف المقروء.

b. مكدس *Stack* له قعر (سنفترضه ممثل بالحرف $) له مؤشر *sp* إلى قمته ويتعامل مع إجرائيتين *Pop* تحذف أعلى القمة وتهبط بالمؤشر إلى القمة الجديدة، وإجرائية *Push(c)* التي تضيف العنصر *c* إلى أعلى المكدس وترفع المؤشر ليشير إليه.

c. جدول القواعد الذي سبق وقمنا ببنائه في الفقرة السابقة.

تكون خوارزمية تحليل جملة *W* على الشكل التالي:

```
Repeat
{
Stack[sp]=S;
X=Stack[sp];
a=W[wp];
if (X is non terminal) then
{
        if (M[X, a]= "X → Y1...Yn ") then
        {
        Pop;
        for (i=n down to 1) do
        Push(Yi)
        }
        else ERROR;
}
else
{
        if (X= =$) then
        if (a = = $) then ACCEPTED
        else ERROR;
        else
        if (X = = a) then { Pop; wp=wp+1; a=W[wp];}
        else ERROR;

}
Until ERROR or ACCEPTED
```

مثال:

لنأخذ الجدول السابق الذي سبق وبنيناه للقواعد:

$$E \to T E'$$
$$E' \to + T E' | \varepsilon$$
$$T \to F T'$$
$$T' \to * F T' | \varepsilon$$
$$F \to (E) | Id$$

	Id	+	*	(	)	$
E	$E \to TE'$			$E \to TE'$		
E′		$E' \to +TE'$			$E' \to \varepsilon$	$E' \to \varepsilon$
T	$T \to FT'$			$T \to FT'$		
T′		$T' \to \varepsilon$	$T' \to *FT'$		$T' \to \varepsilon$	$T' \to \varepsilon$
F	$F \to Id$			$F \to (E)$		

لنحلل الجملة *$a+b*c* حيث *a* و *b* و *c* معرّفات *(Identifiers)*:

Stack	*Input Word*	*Rule*
$E	*a+b*c$*	$E \to TE'$
$ET	*a+b*c$*	$T \to FT'$
$ETF	*a+b*c$*	$F \to Id$
$ETa	*a+b*c$*	
$ET	*+b*c$*	$T' \to \varepsilon$
$E	*+b*c$*	$E' \to +TE'$
$ET+	*+b*c$*	
$ET	*b*c$*	$T \to FT'$

Stack	*Input Word*	*Rule*
$ETF	*b*c$*	$F \rightarrow Id$
$ETb	*b*c$*	
$ET	**c$*	$T' \rightarrow *FT'$
*$ETF**	**c$*	
$ETF	*c$*	$F \rightarrow Id$
$Etc	*c$*	
$ET	*$*	$T' \rightarrow \varepsilon$
$E	*$*	$E' \rightarrow \varepsilon$
$	*$*	*ACCEPTED*

b. النحو *LL(1)*

لا يمكن تطبيق الخوارزمية السابقة على جميع أنواع النحو الصرفي. فإذا وجد في إحدى خانات جدول القواعد عدد من القواعد عوضاً عن قاعدة وحيدة، لن تتمكن الخوارزمية من تحديد أي قاعدة هي القاعدة الصحيحة والتي يجب اعتمادها.

لذا، نعرِّف نحو صرفي بأنه «**نحو صرفي من النمط *LL(1)***" في حال كان جدول القواعد المبني لهذا النحو لا يحوي على خانات فيها عدة قواعد متعددة. ونعني بالمصطلح *LL(1)*:

(المسح من **اليسار** ومن ثم الاشتقاق **اليساري** للقواعد مع توقع حرف **وحيد** إلى الأمام)

*(**L**eft scanning **L**eftmost derivation with **1** look ahead symbol)*

مثال:

النحو التالي ليس نحو من النمط *LL(1)*

$$\boxed{\begin{array}{l} S \rightarrow \text{if E then S } S' \mid b \\ S' \rightarrow \text{else S} \mid \varepsilon \\ E \rightarrow b \end{array}} \rightarrow \begin{cases} \text{First (S)} = \{\text{if, b}\} \\ \text{First (}S'\text{)} = \{\text{else}, \varepsilon\} \\ \text{First (E)} = \{\text{b}\} \\ \text{Follow (S)} = \{\$\} \\ \text{Follow (}S'\text{)} = \{\text{else}, \$\} \\ \text{Follow (E)} = \{\text{then}, \$\} \end{cases} \rightarrow$$

	b	*else*	*if*	*then*	*$*
S	$S \rightarrow b$		$S \rightarrow \text{if E then S } S' \mid b$		
S'		$S' \rightarrow \varepsilon$ $S' \rightarrow else\,S$			$S' \rightarrow \varepsilon$
E	$E \rightarrow b$				

نتيجة: أي نحو صرفي يتصف بإحدى الصفتين التاليتين:

i. عودي يساري *(left recursive)*.

ii. له معاملات يسارية مشتركة ولكنه غير مختصر عبر حساب معاملاته اليسارية المشتركة *(not left factorized)*.

لا يكون من النمط *LL(1)*.

c. العودية اليسارية (Left Recursion)

تعريف: يكون النحو الصرفي "**عودي يساري مباشر**" إذا كان فيه رمز وسيط A له قاعدة صرفية من الشكل:

$$A \rightarrow A\alpha : \alpha \in (\mathrm{V} \cup \mathrm{T})*$$

مثال:

$$\begin{array}{|l|}\hline S \rightarrow ScA \mid B \\ A \rightarrow Aa \mid \varepsilon \\ B \rightarrow Bb \mid d \mid \varepsilon \\ \hline\end{array}$$

نظرية:

لحذف العودية اليسارية المباشرة نطبق الخوارزمية التالية:

$$\forall A : (A \rightarrow Aa \mid \beta) \Leftrightarrow \begin{cases} A \rightarrow \beta A' \\ A' \rightarrow \alpha A' \mid \varepsilon \end{cases}$$

ويكون النحو الناتج مكافئ تماماً للنحو الأصلي حيث يولد نفس اللغة التي يولدها النحو الأصلي

مثال: بتطبيق الخوارزمية السابقة نحصل على:

$$\begin{array}{|l|}\hline S \rightarrow ScA \mid B \\ A \rightarrow Aa \mid \varepsilon \\ B \rightarrow Bb \mid d \mid e \\ \hline\end{array} \rightarrow \begin{array}{|l|}\hline \mathrm{S} \rightarrow \mathrm{BS}' \\ \mathrm{S}' \rightarrow \mathrm{cAS}' \mid \varepsilon \\ \mathrm{A} \rightarrow \mathrm{A}' \\ \mathrm{A}' \rightarrow \mathrm{aA}' \mid \varepsilon \\ \mathrm{B} \rightarrow \mathrm{dB}' \mid \mathrm{eB}' \\ \mathrm{B}' \rightarrow \mathrm{bB}' \mid \varepsilon \\ \hline\end{array}$$

تعريف: يكون النحو الصرفي "**عودي يساري**" إذا كان فيه رمز وسيط A له اشتقاق من الشكل:

$$A \overset{+}{\rightarrow} A\alpha : \alpha \in (V \cup T)*$$

مثال: هذا النحو عودي يساري بسبب الرمز الوسيط S وقاعدته. ولكنه ليس عودياً يسارياً مباشراً:

$$\boxed{\begin{array}{l} S \rightarrow Aa \mid b \\ A \rightarrow Ac \mid Sd \mid \varepsilon \end{array}} \Rightarrow \boxed{S \rightarrow Aa \rightarrow Sda}$$

نظرية:

لحذف العودية اليسارية نطبق الخوارزمية التالية:

Make an order between non terminals: $A_1, ..., A_n$

for *i=1 to n* ***do***

{

  for *j=1 to i-1* ***do***

  {

    Substitute each production of the form $A_i \rightarrow A_j\alpha$ *or* $A_j \rightarrow \beta_1 \mid ... \mid \beta_p$ *by a production of the form* $A_i \rightarrow \beta_1\alpha \mid ... \mid \beta_p\alpha$

  }

  Eliminate the direct left recursion of productions of A_i

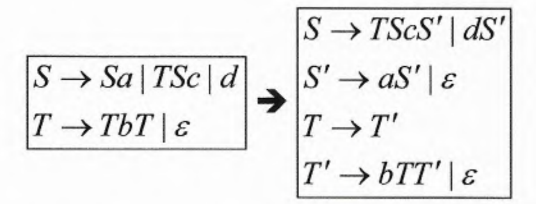
}

ويكون النحو الناتج مكافئ تماماً للنحو الأصلي حيث يولد نفس اللغة التي يولدها النحو الأصلي.

مثال *1*:

$$\begin{array}{|l|}\hline S \to Aa \mid b \\ A \to Ac \mid Sd \mid \varepsilon \\ \hline \end{array} \;\rightarrow\; \begin{array}{|l|}\hline S \to Aa \mid b \\ A \to bdA' \mid A' \\ A' \to cA' \mid adA' \mid \varepsilon \\ \hline \end{array}$$

مثال *2*: لنأخذ المثال التالي



لكننا نلاحظ هنا أنه وبعد تطبيق الخوارزمية يمكن أن نحصل على عودية يسارية غير مباشرة:

$$\boxed{S \to TScS' \to T'ScS' \to ScS'}$$

يعود السبب في ذلك إلى وجود قاعدة $T' \to \varepsilon$.

> **نتيجة:** الخوارزمية لا تعمل بشكل صحيح إذا وجدت قواعد من النمط $A \to \varepsilon$.

d. حساب المعاملات اليسارية المشتركة

عندما يكون على خوارزمية التحليل اتخاذ قرار بين عدد من القواعد، تساعد عملية حساب المعاملات اليسارية المشتركة في تأخير اتخاذ القرار ريثما تكون الجملة قد أصبحت أكثر اكتمالاً ما يسمح باتخاذ قرار أفضل.

مثال: لتكن لدينا القواعد الصرفية التالية:

$$
\begin{array}{|l|}
\hline
S \rightarrow bacdAbd \mid bacdBcca \\
A \rightarrow aD \\
B \rightarrow cE \\
C \rightarrow \ldots \\
E \rightarrow \ldots \\
\hline
\end{array}
$$

في هذه الحالة، نجد أن لدينا خيارين من أجل S، ولمعرفة الخيار الصائب، يتوجب علينا قراءة *4* رموز والوصول إلى الرمز الخامس، لنعرف إذا كنا سنحصل على a (حالة الرمز الوسيط A) أو سنحصل على c (حالة الرمز الوسيط B). إذاً لا يمكن لنا منذ البداية معرفة القاعدة التي يتوجب علينا اختيارها وهو ما يتعارض مع مفهو النحو الصرفي من النمط $LL(1)$.

نظرية: يمكن حساب المعاملات المشتركة (في حال وجدت) في نحو الصرفي اعتماداً على الخوارزمية:

***for** each non terminal A **do** {*

Find the common prefix $\alpha \neq \varepsilon$ *and substitute:* $A \rightarrow \alpha B_1 \mid \ldots \mid \alpha B_n \mid \gamma_1 \mid \ldots \mid \gamma_P$ *by the two following rules:*

$$
\begin{cases}
A \rightarrow \alpha A' \mid \gamma_1 \mid \ldots \mid \gamma_P \\
A' \rightarrow B_1 \mid \ldots \mid B_n
\end{cases}
$$

}

مثال:

$$\begin{array}{|l|}\hline S \to aEbS \mid aEbSeB \mid a \\ E \to bcB \mid bca \\ B \to ba \\ \hline \end{array} \Rightarrow \begin{array}{|l|}\hline S \to aEbSS' \mid a \\ S' \to eB \mid \varepsilon \\ E \to bcE' \\ E' \to B \mid a \\ B \to ba \\ \hline \end{array}$$

e. النحو النظيف

نقول عن نحو صرفي أنه **"نحو نظيف"** إذا كان خالياً من قواعد من الشكل: $A \to \varepsilon$. يمكن تحويل نحو صرفي إلى نحو صرفي نظيف إذا

قمنا من أجل كل رمز A يمتلك قاعدة من الشكل $A \to \varepsilon$ ، باستبدل

كل ظهور له ضمن الجهة اليمينية من أي قاعدة من القواعد الصرفية بالرمز ε .

مثال:

$$\begin{array}{|l|}\hline S \to aTb \mid aU \\ T \to bTaTA \mid \varepsilon \\ U \to aU \mid b \\ \hline \end{array} \Rightarrow \begin{array}{|l|}\hline S \to aTb \mid ab \mid aU \\ T \to bTaTA \mid baTA \mid bTaA \mid baA \\ U \to aU \mid b \\ \hline \end{array}$$

f. استنتاج

يمكن أتمتة عملية التحليل النازل، لنحو صرفي من النوع $LL(1)$ على أن يكون النحو:

1. غير غامض وهو أمر يعود إلى مصمم النحو.

2. لا يحوي على عودية يسارية بحيث تتم إزالة العودية اليسارية *(إذا وجدت)* باستخدام خوارزمية التحويل إلى نحو دون عودية يسارية.

3. مختصر بعد حساب معاملاته اليسارية المشتركة *(إذا وجدت)* وذلك باستخدام خوارزمية التحويل.

4. بناء جدول القواعد بعد حساب مجموعتي الرموز الأولى والرموز اللاحقة *(First & Follow)* وفق خوارزميات الحساب المناسبة.

5. تنفيذ عملية التحليل باستخدام المكدس وفق خوارزمية التحليل النازل.

مثال:

1. اعتباراً من نحو غامض: $\boxed{E \rightarrow E+E \mid E*E \mid (E) \mid Id}$

2. يجب هيكلته باستخدام الخواص التجميعية للعمليات الحسابية ليصبح:

$$\begin{cases} E \rightarrow E+T \mid T \\ T \rightarrow T*F \mid F \\ F \rightarrow (E) \mid nb \end{cases}$$

3. يتم تطبيق خوارزمية إلغاء العودية اليسارية لنحصل على:

$$\begin{cases} E \rightarrow TE' \\ E' \rightarrow +TE' \mid \varepsilon \\ T \rightarrow FT' \\ T' \rightarrow *FT' \mid \varepsilon \\ F \rightarrow (E) \mid Id \end{cases}$$

4. نلاحظ أن هذا النحو هو من النمط $LL(1)$ وليس عودياً من اليسار ولا حاجة لاختصاره بحساب العاملات اليسارية المشتركة لأنها محسوبة. لذا يمكن تطبيق خوارزمية التحليل مباشرةً عليه.

1 - 7 - 2 - تحليل صاعد من الأسفل إلى الأعلى

يعتمد التحليل الصاعد في مبدأ عمله على بناء شجرة الاشتقاق من الأسفل (من الأوراق التي تمثل الرموز الأولية مروراً بالعقد التي تمثل الرموز الوسيطة) إلى الأعلى (رمز البداية).

يدعى النموذج العام المستخدم بنموذج **السحب / الاختصار (*Shift/Reduce*)** ويقوم هذا النموذج على عمليتين هما:

1. **السحب (*Shift*):** التي تعني سحب المؤشر المار على الجملة المقروءة بمقدار كلمة إلى الأمام.

2. **الاختصار (*Reduce*):** التي تعني اختصار الجملة من خلال تعويض مجموعة من الرموز التي يمكن أن تشكل سلسلة يمينية لقاعدة صرفية برمز وحيد هو الرمز الظاهر في الناحية اليسارية من هذه القاعدة.

مثال: لنأخذ القاعدة: $S \rightarrow aSbS \mid c$ ولنأخذ الجملة:

u=aaacbaacbcbcbcbacbc

- لنبدأ من الحرف الأول: ***a***_aacbaacbcbcbcbacbc_ حيث لا يمكن أن نختصر **فنسحب**.

- نسحب باتجاه الحرف التالي: _a_***a***_acbaacbcbcbcbacbc_ حيث لا يمكن أن نختصر **فنسحب**.

- نسحب باتجاه الحرف التالي: _aa_***a***_cbaacbcbcbcbacbc_ حيث لا يمكن أن نختصر **فنسحب**.

- نسحب باتجاه الحرف التالي: $aaa\underline{c}baacbcbcbcbacbc$ حيث يمكن أن نختصر وفق $S \to c$.
- نختصر فنحصل على: $aaa\underline{S}baacbcbcbcbacbc$ حيث لا يمكن أن نختصر فنسحب.
- نسحب باتجاه الحرف التالي: $aaaS\underline{b}aacbcbcbcbacbc$ حيث لا يمكن أن نختصر فنسحب.
- نتابع عمليات السحب ....
- نسحب باتجاه الحرف التالي: $aaaSbaa\underline{c}bcbcbcbacbc$ حيث يمكن أن نختصر وفق $S \to c$.
- نختصر فنحصل على: $aaaSbaa\underline{S}bcbcbcbacbc$ حيث لا يمكن أن نختصر فنسحب.
- نسحب باتجاه الحرف التالي: $aaaSbaaS\underline{b}cbcbcbacbc$ حيث لا يمكن أن نختصر فنسحب.
- نسحب باتجاه الحرف التالي: $aaaSbaaSb\underline{c}bcbcbacbc$ حيث يمكن أن نختصر وفق $S \to c$.
- نختصر فنحصل على: $aaaSbaaSb\underline{S}bcbcbacbc$ حيث يمكن أن نختصر وفق $S \to aSbS$.
- نختصر فنحصل على: $aaaSba\underline{S}bcbcbacbc$ حيث لا يمكن أن نختصر فنسحب.
- نتابع عمليات السحب ....
- نسحب باتجاه الحرف التالي: $aaaSbaSb\underline{c}bcbacbc$ حيث يمكن أن نختصر وفق $S \to c$.

- نختصر فنحصل على: $aaSbaSb\underline{\mathbf{S}}bcbacbc$ حيث يمكن أن **نختصر وفق** $S \to aSbS$.

- نختصر فنحصل على: $aaSb\underline{\mathbf{S}}bcbacbc$ حيث يمكن أن **نختصر وفق** $S \to aSbS$.

- نختصر فنحصل على: $aa\underline{\mathbf{S}}bcbacbc$ حيث لا يمكن أن نختصر **فنسحب**.

- نسحب باتجاه الحرف التالي: $aaS\underline{\mathbf{b}}cbacbc$ حيث لا يمكن أن نختصر **فنسحب**.

نسحب باتجاه الحرف التالي: $aaSb\underline{\mathbf{c}}bacbc$ حيث يمكن أن **نختصر وفق** $S \to c$.

- نسحب باتجاه الحرف التالي: $aaSb\underline{\mathbf{S}}bacbc$ حيث يمكن أن **نختصر وفق** $S \to aSbS$.

- نختصر فنحصل على: $a\underline{\mathbf{S}}bacbc$ حيث لا يمكن أن نختصر **فنسحب**.

- نسحب باتجاه الحرف التالي: $aS\underline{\mathbf{b}}acbc$ حيث لا يمكن أن نختصر **فنسحب**.

- نسحب باتجاه الحرف التالي: $aSb\underline{\mathbf{a}}cbc$ حيث لا يمكن أن نختصر **فنسحب**.

- نسحب باتجاه الحرف التالي: $aSba\underline{\mathbf{c}}bc$ حيث يمكن أن **نختصر وفق** $S \to c$.

- نختصر فنحصل على: $aSba\underline{\mathbf{S}}bc$ حيث لا يمكن أن نختصر **فنسحب**.

- نسحب باتجاه الحرف التالي: $aSbaS\underline{\mathbf{b}}c$ حيث لا يمكن أن نختصر **فنسحب**.

- نسحب باتجاه الحرف التالي: $aSbaSb\underline{S}$ حيث يمكن أن **نختصر** وفق $S \rightarrow aSbS$.

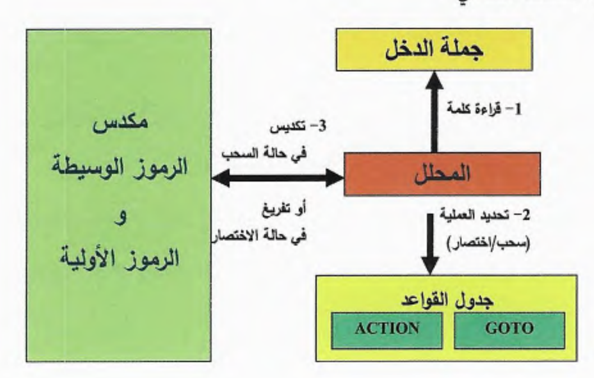
- نختصر فنحصل على: $aSb\underline{S}$ حيث يمكن أن **نختصر** وفق $S \rightarrow aSbS$.

نختصر فنحصل على $\underline{S}$ ونصل إلى رمز البداية، مما يعني أن الجملة التي قمنا بتحليلها صحيحة.

بالنتيجة، سيكون من الضروري الحصول على جدول يساعدنا في معرفة العملية التي يجب تطبيقها، وهل هي عملية سحب أم عملية اختصار.

a. خوارزمية التحليل *LR*:

ستسمح هذه الطريقة بتحليل الجمل التي تنتمي لنحو صرفي من النمط *LR (Leftmost scanning Rightmost derivation)*. ويكون الجدول في هذه الحالة عبارة عن أوتومات ذات مكدس بحيث يسمح لنا في كل مرة نقوم فيها بقراءة حرف، معرفة فيما إذا كانت العملية التي يجب تطبيقها هي عملية سحب او عملية اختصار. تتم عملية التحليل وفق المخطط التالي:



الخوارزمية: سنتعرف على خوارزمية التحليل من خلال مثال. ليكن لدينا النحو الصرفي التالي بعد ترقيم قواعده والجملة التي نريد تحليلها:

$$\begin{cases} (1)E \to E+T \\ (2)E \to T \\ (3)T \to T*F \\ (4)T \to F \\ (5)F \to (E) \\ (6)F \to Id \end{cases}$$

ولنفرض أن جملة الدخل التي نريد تحليلها هي: *[a * b + c $]* والتي تنتهي برمز نهاية الجملة *$*. ولنفرض أن جدول القواعد على النحو التالي (سنتعرف على طريقة بناء الجدول لاحقاً):

	ACTION						GOTO		
	Id	+	*	(	)	$	*E*	*T*	*F*
0	S 5			S 4			1	2	3
1		S 6				Acc			
2		R 2	R 7		R 2	R 2			
3		R4	R 4		R 4	R 4			
4	S 5			S 4			8	2	3
5		R 6	R 6		R 6	R 6			
6	S 5			S 4				9	3
7	S 5			S 4					10
8		S 6			S 11				
9		R 1	S 7		R 1	R 1			
10		R 3	R 3		R 3	R 3			
11		R 5	R 5		R 5	R 5			

- يكون المكدس في الحالة الابتدائية حاوياً على رمز قعر المكدس $ يليه رمز بداية النحو (*E* في مثالنا) ويليه رقم الحالة الأولى (*0* في مثالنا). يكون مؤشر أعلى المكدس مؤشراً على هذه الحالة أي أننا نفترض أننا في الحالة الابتدائية (رقم *0* في حالتنا)
- يكون مؤشر بداية الجملة يؤشر إلى الكلمة الأولى منها (*a* في مثالنا).
- يحتوي العمود اليساري للجدول الحالة s_i الحالية. كما يحتوي السطر العلوي رموز النحو الصرفي.
- اعتباراً من الحالة s_i اقرأ الكلمة a_j من الجملة (الكلمة التي يؤشر إليها مؤشر بداية الجملة):
  - إذا كان محتوى الخانة $Action[s_i,a_j]=Sk$ أي عملية سحب مرفقة بالرقم *k* يكون:
    - ادفع الرمز *aj* إلى أعلى المكدس وادفع وراءه بالحالة *si*.
    - تقدم كلمة واحدة إلى الأمام في الجملة.
  - إذا كان محتوى الخانة *kR=[ja,is]noitcA* أي عملية اختصار مرفقة بالرقم *k* يكون:
    - اسحب من أعلى المكدس خانات عددها: *2**(طول السلسلة اليمينية من القاعدة رقم *k*).
    - ادفع إلى أعلى المكدس بمحتوى $GOTO[s_m,a_n]$ حيث a_n هو الرمز اليساري للقاعدة رقم *k*، وs_m هو الرمز الموجود في أعلى المكدس بعد تنفيذ عملية التفريغ السابقة.
- نكرر العمل طالما لم نصل إلى نهاية الجملة.

b. بناء جدول التحليل

سنبني جدول التحليل أيضاً اعتماداً على المثال الذي استخدمنه في الفقرة السابقة.

ليكن لدينا النحو الصرفي:

$$\begin{cases} E \rightarrow E + T \\ E \rightarrow T \\ T \rightarrow T * F \\ T \rightarrow F \\ F \rightarrow (E) \\ F \rightarrow Id \end{cases}$$

سنفترض الآن أننا سنتعامل مع القواعد بصيغة الحالات وأن لدينا حالة ابتدائية تحتوي على جميع القواعد مع مؤشر لقراءة الرموز نرمز له بنقطة (•) فيكون للقواعد الشكل التالي وندعوها الحالة I_0:

$$I_0 = \begin{cases} E \rightarrow \bullet E + T \\ E \rightarrow \bullet T \\ T \rightarrow \bullet T * F \\ T \rightarrow \bullet F \\ F \rightarrow \bullet (E) \\ F \rightarrow \bullet Id \end{cases}$$

أولاً- الإغلاق على مجموعة من القواعد التي تشكل حالة:

إذا كان لدينا مجموعة من القواعد التي تشكل حالة ما *I*، فيمكن حساب الإغلاق على هذه الحالة، والذي ندعوه *Closure(I)* وفقاً للخوارزمية التالية:

a. نضع كل عنصر من الحالة *I* ضمن *Closure(I)*.

b. من أجل كل عنصر من *Closure(I)* له الشكل:

$$A \rightarrow \alpha \bullet B\beta$$

- من أجل كل قاعدة $B \rightarrow \gamma_j$
  - نضع هذه القاعدة في *Closure(I)*.

c. كرر العمل حتى يتم إغلاق *Closure(I)* ولا يعود هناك ما نضيفه.

فمثلاً، اعتباراً من النحو السابق ومن $I = \begin{cases} E \rightarrow E \bullet +T \\ T \rightarrow T * \bullet F \end{cases}$

يكون لدينا $Closure(I) = \begin{cases} E \rightarrow E \bullet +T \\ T \rightarrow T * \bullet F \\ F \rightarrow \bullet(E) \\ F \rightarrow \bullet Id \end{cases}$.

ثانياً - تعريف الانتقال من حالة إلى حالة:

نعرف الانتقال من حالة I_j إلى حالة أخرى I_k وذلك بعد قراءة رمز *X* كما يلي:

$$I_k = \Delta(I_j, X) = Closure(\{A \rightarrow \alpha X \bullet \beta \mid [A \rightarrow \alpha \bullet X\beta] \in I_j\})$$

فمثلاً، تكون نتيجة الانتقال من $I_j = \begin{cases} E \rightarrow E \bullet +T \\ T \rightarrow T * \bullet F \\ F \rightarrow \bullet(E) \\ F \rightarrow \bullet Id \end{cases}$

بقراءة الرمز *F* هي: $I_k = \{T \rightarrow T * F \bullet\}$؛

في حين تكون نتيجة الانتقال من $I_j = \begin{cases} E \to E \bullet +T \\ T \to T * \bullet F \\ F \to \bullet(E) \\ F \to \bullet Id \end{cases}$

بقراءة الرمز + هي: $I_m = \begin{cases} E \to E + \bullet T \\ T \to \bullet T * F \\ T \to \bullet F \\ F \to \bullet(E) \\ F \to \bullet Id \end{cases}$.

ثالثاً - بناء حالات التحليل (الحالات التي وضعنا أرقامها في العمود اليساري من الجدول):

d. أضف القاعدة $S' \to S$ (حيث *S* هو رمز البداية).

e. أحسب $I_0 = Closure(S' \to \bullet S)$.

f. أضف I_0 إلى مجموعة الحالات.

g. من أجل كل حالة I من مجموعة الحالات:

- من أجل كل رمز *X* من رموز النحو حيث $\Delta(I, X) \neq \phi$:
- أضف $\Delta(I, X)$ إلى مجموعة الحالات.

رابعاً - بناء الجدول:

h. ترقيم القواعد الصرفية.

i. بناء مجموعة الحالات $\{I_0,...,I_n\}$.

j. من أجل كل حالة $I_j = \Delta(I_i, a)$ حيث a رمز أولي:

- $\text{Action}[i,a] = \text{Sj}$

k. من أجل كل حالة $I_j = \Delta(I_i, A)$ حيث A رمز وسيط:

- $\text{GoTo}[i,A] = j$

l. من أجل كل حالة $A \to \alpha \bullet$ محتواة في I_i:

- من أجل كل $a \in \text{Follow}(A)$:
  - $\text{Action}[i,a] = \text{Rk}$ حيث k هي رقم القاعدة ضمن لائحة القواعد الصرفية.

الأخطاء الصرفية *(Syntax Errors)*

هناك العديد من الأخطاء التي قد تأتي من عدم كتابة المفردات بالشكل الصحيح *(أخطاء تحليل مفرداتي)*، أو من عدم تركيب الجمل بالشكل الصحيح *(أخطاء صرفية)*. بجميع الأحوال، يجب على معالج الأخطاء أن:

- يحدد وجود الخطأ بشكل واضح ودقيق.
- يعالج الخطأ بسرعة حرصاً على سرعة استكمال التحليل.
- يعالج الخطأ بطريقة فعالة دون أن يؤدي الأمر لتوليد خطأ جديد.

لحسن الحظ، من السهل معالجة الأخطاء المعتادة الناجمة عن فقدان حرف أو كلمة أو استبدال كلمة بأخرى (مثل فقدان فاصلة منقوطة، استخدام فاصلة عادية عوضاً عن فاصلة منقوطة أو بالعكس، ...)، وهي أخطاء لا تحتاج لآليات معقدة لمعالجتها. بكل الأحوال، يمكن لمثل هذه الأخطاء أن تكون موجودة قبل فترة طويلة من اكتشافها. فعلى سبيل المثال لا يظهر فقدان القوس المخصص لبداية أو نهاية مقطع إلا عند نهاية المقطع، مما يجعل من الصعب اكتشاف مصدر الخطأ فيصبح برنامج التحليل مضطراً "لتوقع" ما يوجد في رأس المبرمج الذي كتب البرنامج.

على كل حال توجد استراتيجيات متعددة لمعالجة الأخطاء، ولكن من المهم الانتباه إلى ضرورة إيقاف التحليل عند تجميع عدد كبير من الأخطاء. إذ يصبح عندها من غير المفيد الاستمرار كون الأخطاء التي سيتم اكتشافها لاحقاً ستكون نتيجة للأخطاء السابقة. وعليه هناك عدة استراتيجيات لمتابعة ومعالجة الأخطاء، وهي الحالات التي نستعرضها فيما يلي:

a. أسلوب *Panic Mode*

وهو الأبسط في المعالجة، إذ يقوم المحلل، عند مواجهته لخطأ ما، بمتابعة القراءة من خلال حذف رموز الدخل الواحد تلو الآخر حتى يصل إلى إحدى المفردات الأساسية التي تشكل نهاية جملة أو مقطع (مثل "؛" أو "*end*" أو "}" ...) والتي يكون لها دور واضح في البرنامج المصدري الذي يجري تحليله. ومن هناك يبدأ من جديد مع تحديد وجود خطأ في بداية المقطع السابق. يتميز هذا الأسلوب ببساطته، ولكن نتائج الأخطاء التي يعطيها تكون غير مفصلة وغير دقيقة وتترك قسماً كبيراً من النص المصدري دون تحقق.

b. تصحيح الأخطاء

يمكن في بعض الحالات محاولة تصحيح بعض الأخطاء. فعند تنفيذ التحليل القواعدي باستخدام خوارزميات التحليل التي سبق وذكرناها، يمكننا واعتماداً على الجداول، محاولة أتمتة عملية توقع خطأ. ففي جميع الحالات التي درسناها، يمتلك جدول القواعد (أو جدول التحليل) خانات فارغة، ويكون وقوع خوارزمية التحليل على إحداها معبراً عن اكتشاف خطأ ما. مما يسمح لنا بتوقع الحالات التي يمكن أن تقع فيها خوارزمية التحليل على خانة فارغة وتوقع تصحيحها.

إلا أن سيئة هذه العملية تكمن في أن العديد من الأخطاء التي قد تسبب وقوع خوارزمية التحليل في خانة فارغة، قد تكون ناشئة أصلاً عن خطأ يسبق بكثير الوضع الحالي، مما يعني أن أي تصحيح قد يكون بدوره خاطئاً.

c. إضافة قواعد للأخطاء

يمكننا معالجة العديد من الأخطاء الشائعة في حال كان لدينا فكرة واضحة عنها، عبر إضافة قواعد تحدد الخطأ. فعلى سبيل المثال عند وجود جملة شرطية مثل:

if (Expression) then INST else INST

يمكننا إضافة قاعدة خطأ مثل:

I à if Expression then (Error – No Parenthesis).

1 - 7 - 3 - مسألة

باستخدام النحو الصرفي G سنقوم بتوصيف مجموعة جزئية من اللغة الإنكليزية.

- سنفرض أن لدينا مجموعة الرموز المنتهية *(Terminals)* التالية:

$$T=\{Verb, Name, who, and\}$$

  - يأخذ *Verb* إحدى القيم التالية:

$$Verb = \{ loves, follows, slaps, \ listen\ to, deranges\}$$

  - يأخذ*Name* إحدى القيم التالية:

$$Name=\{Marc, Brad, Bob, Anne, Sophie\}$$

- لا تأخذ الرموز *{who, and}* إلا قيمة واحدة هي قيمتها نفسها.

لنفرض أن الرموز الوسيطة *(Non Terminals)* في هذا النحو هي:

$$N=\{P, S, C, R\}:$$

- يمثل الرمز P الجملة *(Sentence)*؛
- يمثل الرمز S الفاعل *(Subject)*؛
- يمثل الرمز C المفعول به *(Complement)*؛
- يمثل الرمز R حروف الوصل *(Subordinates)*.

تكون القواعد الصرفية الخاصة بهذا النحو على الشكل التالي:

(1) $P \rightarrow S$ *verb* C
(2) $S \rightarrow$ *Name*
(3) $C \rightarrow$ *Name*
(4) $C \rightarrow$ *Name* R
(5) $R \rightarrow$ *who verb* C
(6) $R \rightarrow R$ *and* R

الأسئلة:

1. نريد تحليل الجملة التالية (بغض النظر عن تصريف الأفعال):

« Marc follows Bob who listen to Anne who deranges Brad and who loves Sophie »

a. أعط شجرة اشتقاق خاصة بهذه الجملة.

b. برأيك هل النحو المقترح غامض؟ علل إجابتك.

2. عدل القواعد لإلغاء العودية اليسارية وإلغاء المعاملات المشتركة.

3. لندعو $G1$ النحو الصرفي الناتج بعد إلغاء العودية اليسارية والمعاملات المشتركة.

a. قم ببناء منظومة التحليل *Top-Down Parsing* من النمط *LL(1)* للنحو $G1$ من خلال بناء جدول القواعد (أو جدول المناقلات *Production Table*).

4. اعتباراً من النحو G سنقوم بإلغاء القاعدة الأخيرة للرمز R (القاعدة رقم 6) لنحصل على النحو $2G$:

a. قم ببناء منظومة تحليل *Bottom-Up Parsing* للنحو $G2$ من خلال بناء جداول *ACTION* و*GOTO*.

الجـواب الأول:

« Marc follows Bob who listen to Anne who deranges Brad and who loves Sophie »

a. نطبق عملية اشتقاق يساري للقواعد:

P →

S Verb C →

Marc Verb C →

Marc follows C →

Marc follows Name R →

Marc follows Bob R →

Marc follows Bob who Verb C →

Marc follows Bob who listen to C →

Marc follows Bob who listen to Name R →

Marc follows Bob who listen to Anne R →

Marc follows Bob who listen to Anne R and R →

Marc follows Bob who listen to Anne who verb C and R →

Marc follows Bob who listen to Anne who deranges C and R →

Marc follows Bob who listen to Anne who deranges Name and R →

Marc follows Bob who listen to Anne who deranges Brad and R →

Marc follows Bob who listen to Anne who deranges Brad and who Verb C →

Marc follows Bob who listen to Anne who deranges Brad and who loves C →

Marc follows Bob who listen to Anne who deranges Brad and who loves Sophie

b. نلاحظ أن القواعد غامضة والبرهان وجود مسار اشتقاق يساري آخر لنفس الجملة:

P →

S Verb C →

Marc Verb C →

Marc follows C →

Marc follows Name R →

Marc follows Bob R and R → هنا يبدأ المسار المختلف

Marc follows Bob who Verb C and R →

Marc follows Bob who listen to C and R →

Marc follows Bob who listen to Name and R →

Marc follows Bob who listen to Anne R and R →

Marc follows Bob who listen to Anne R and R →

Marc follows Bob who listen to Anne who verb C and R →

Marc follows Bob who listen to Anne who deranges C and R →

Marc follows Bob who listen to Anne who deranges Name and R →

Marc follows Bob who listen to Anne who deranges Brad and R →

Marc follows Bob who listen to Anne who deranges Brad and who Verb C →

Marc follows Bob who listen to Anne who deranges Brad and who loves C →

Marc follows Bob who listen to Anne who deranges Brad and who loves Sophie

الجواب الثاني:

نطبق خوارزمية إلغاء العودية اليسارية المباشرة على القاعدة الوحيدة التي تحتويها:

$$\begin{array}{|l|}\hline P \rightarrow S\ \text{Verb}\ C \\ S \rightarrow \text{Name} \\ C \rightarrow \text{Name} \mid \text{Name}\ R \\ R \rightarrow \text{who Verb}\ C\ R' \\ R' \rightarrow \text{and}\ R\ R' \mid \varepsilon \\ \hline \end{array} \quad \Rightarrow \quad \begin{array}{|l|}\hline P \rightarrow S\ \text{Verb}\ C \\ S \rightarrow \text{Name} \\ C \rightarrow \text{Name} \mid \text{Name}\ R \\ R \rightarrow R\ \text{and}\ R \mid \text{wh}\ \ \text{Verb}\ C \\ \hline \end{array}$$

نطبق خوارزمية إلغاء العودية اليسارية (غير المباشرة):

$$\begin{array}{|l|}\hline P \rightarrow S\ \text{Verb}\ C \\ S \rightarrow \text{Name} \\ C \rightarrow \text{Name} \mid \text{Name}\ R \\ R \rightarrow \text{who Verb}\ C\ R' \\ R' \rightarrow \text{and}\ R\ R' \mid \varepsilon \\ \hline \end{array} \quad \Rightarrow \quad \begin{array}{|l|}\hline P \rightarrow \text{Name Verb}\ C \\ C \rightarrow \text{Name} \mid \text{Name}\ R \\ R \rightarrow \text{who Verb}\ C\ R' \\ R' \rightarrow \text{and}\ R\ R' \mid \varepsilon \\ \hline \end{array}$$

نطبق خوارزمية حساب المعاملات المشتركة (في حال وجدت) في نحو الصرفي:

$$\begin{array}{|l|}\hline P \rightarrow \text{Name Verb}\ C \\ C \rightarrow \text{Name} \mid \text{Name}\ R \\ R \rightarrow \text{who Verb}\ C\ R' \\ R' \rightarrow \text{and}\ R\ R' \mid \varepsilon \\ \hline \end{array} \quad \Rightarrow \quad \begin{array}{|l|}\hline P \rightarrow \text{Name Verb}\ C \\ C \rightarrow \text{Name}\ C' \\ C' \rightarrow R \mid \varepsilon \\ R \rightarrow \text{who Verb}\ C\ R' \\ R' \rightarrow \text{and}\ R\ R' \mid \varepsilon \\ \hline \end{array}$$

الجواب الثالث:

نقوم بحساب *First(X)* من أجل كل رمز وسيط *X* ينتمي إلى النحو الصرفي:

$$P \rightarrow Name\ Verb\ C$$
$$C \rightarrow Name\ C'$$
$$C' \rightarrow R \mid \varepsilon$$
$$R \rightarrow who\ Verb\ C\ R'$$
$$R' \rightarrow and\ R\ R' \mid \varepsilon$$

➔

$$First(P) = First(C) = \{Name\}$$
$$First(C') = \{who, \varepsilon\}$$
$$First(R) = \{who\}$$
$$First(R') = \{and, \varepsilon\}$$

نقوم بحساب *Follow(X)* من أجل كل رمز وسيط *X* ينتمي إلى النحو الصرفي:

$$P \rightarrow Name\ Verb\ C$$
$$C \rightarrow Name\ C'$$
$$C' \rightarrow R \mid \varepsilon$$
$$R \rightarrow who\ Verb\ C\ R'$$
$$R' \rightarrow and\ R\ R' \mid \varepsilon$$

➔

$$Follow(P) = Follow(R) = Follow(R') = \{\$\}$$
$$Follow(C) = Follow(C') = \{\$\ and\}$$

نقوم ببناء الجدول:

	Name	*Verb*	*who*	*and*	*$*
P	$P \rightarrow Name\ Verb\ C$				
C	$C \rightarrow Name\ C'$				
C′			$C' \rightarrow R$	$C' \rightarrow \varepsilon$	$C' \rightarrow \varepsilon$
R			$R \rightarrow who\ Verb\ C\ R'$		
R′				$R' \rightarrow and\ R\ R'$	$R' \rightarrow \varepsilon$

الجواب الرابع:

يكون النحو $G2$ على النحو التالي:

$$\begin{array}{l} P \rightarrow S\ Verb\ C \\ S \rightarrow Name \\ C \rightarrow Name \mid Name\ R \\ R \rightarrow who\ Verb\ C \end{array}$$

لبناء الجدول، نبدأ بترقيم القواعد الصرفية وفقاً للخوارزمية المعتمدة لبناء جدول التحليل:

$$\begin{array}{l} (1)\ P \rightarrow S\ Verb\ C \\ (2)\ S \rightarrow Name \\ (3)\ C \rightarrow Name \\ (4)\ C \rightarrow Name\ R \\ (5)\ R \rightarrow who\ Verb\ C \end{array}$$

لنبدأ ببناء الحالات وفقاً لخوارزمية توليد الحالات حيث نعتبر أن قاعدة P هي القاعدة الابتدائية كونها وحيدة ونحسب الإغلاق على P:

$$I_0 = \begin{cases} P \rightarrow \bullet S\ Verb\ C \\ S \rightarrow \bullet Name \end{cases}$$

$$I_0 = \begin{cases} P \rightarrow \bullet S\ Verb\ C \\ S \rightarrow \bullet Name \end{cases} \Rightarrow \begin{array}{l} I_1 = \Delta(I_0, S) = \{P \rightarrow S \bullet Verb\ C \\ I_2 = \Delta(I_0, Name) = \{S \rightarrow Name \bullet \end{array}$$

$$I_1 = \{P \rightarrow S \bullet Verb\ C\} \Rightarrow I_3 = \Delta(I_1, Verb) = \begin{cases} P \rightarrow S\ Verb \bullet C \\ C \rightarrow \bullet Name\ R \\ C \rightarrow \bullet Name \end{cases}$$

$$I_3 = \begin{cases} P \rightarrow S\ Verb \bullet C \\ C \rightarrow \bullet Name\ R \\ C \rightarrow \bullet Name \end{cases} \Rightarrow \begin{array}{l} I_4 = \Delta(I_3, C) = \{P \rightarrow S\ Verb\ C \bullet \\ I_5 = \Delta(I_3, Name) = \begin{cases} C \rightarrow Name \bullet \\ C \rightarrow Name \bullet R \\ R \rightarrow \bullet who\ Verb\ C \end{cases} \end{array}$$

$$I_5 = \begin{cases} C \rightarrow \text{Name} \bullet \\ C \rightarrow \text{Name} \bullet R \\ R \rightarrow \bullet \text{who Verb C} \end{cases} \Rightarrow \begin{array}{l} I_6 = \Delta(I_5, R) = \{C \rightarrow \text{Name R} \bullet \\ I_7 = \Delta(I_5, \text{who}) = \{C \rightarrow \text{who} \bullet \text{Verb C} \end{array}$$

$$I_7 = \{C \rightarrow \text{who} \bullet \text{Verb C} \Rightarrow \Delta(I_7, \text{Verb}) = I_3$$

لنحسب المجموعات *Follow(X)* حيث *X* هو رمز وسيط من النحو الصرفي:

$$\begin{array}{l} (1)\ P \rightarrow \text{S Verb C} \\ (2)\ S \rightarrow \text{Name} \\ (3)\ C \rightarrow \text{Name} \\ (4)\ C \rightarrow \text{Name R} \\ (5)\ R \rightarrow \text{wh Verb C} \end{array} \Rightarrow \begin{array}{l} \text{Follow}(P) = \{\$\} \\ \text{Follow}(S) = \{\text{Verb}\} \\ \text{Follow}(C) = \{\$\} \\ \text{Follow}(R) = \{\text{who},\$\} \end{array}$$

ويكون الجدول بعد تطبيق خوارزمية بنائه كما يلي:

	ACTION				GOTO			
	Name	Verb	who	$	P	S	C	R
0	S2					1		
1		S3						
2		R2						
3	S5						4	
4			R1	R1				
5			S7	R3				6
6			R4	R4				
7		S3						

1 - 8 - التحليل الدلالي

هناك مجموعة من خصائص لغات البرمجة التي لا يمكن توصيفها باستخدام النحو خارج السياق الذي استخدمناه لتوصيف القواعد الصرفية للغة، لأنها وببساطة خصائص ترتبط ارتباط مباشر بسياق البرنامج المكتوب بهذه اللغة. فعلى سبيل المثال لا الحصر: لا يمكننا الإعلان عن متحول مرتين في نفس الإجرائية أو المقطع البرمجي (سنعرف مفهوم المقطع لاحقاً)، كما لا يمكننا استخدام متحول دون الإعلان عنه بشكل مسبق (في بعض لغات البرمجة). كما لا يمكننا أن نبرمج عملية جداء عدد حقيقي بسلسلة محارف.

يهتم المحلل الدلالي (أو محلل السياق) بالتحقق من الخصائص المرتبطة بالسياق للغة البرمجة، ويجري تنفيذ عملية التحليل الدلالي في نفس الوقت الذي تتم فيه عملية التحليل القواعدي الصرفي وذلك اعتماداً على إجرائيات وعمليات يتم استدعاؤها ضمن القواعد الصرفية.

عموماً، لا توجد طريقة وأسلوب وبنى محددة لتنفيذ التحليل الدلالي. فالعملية ترتبط من ناحية، بلغة البرمجة وقواعدها، وترتبط من ناحية أخرى بأسلوب مصمم المترجم وطريقة تصميمه لبنى المعطيات التي ستستقبل معلومات السياق (مثل المتحولات وأنماطها عند الإعلان عنها)، والإجرائيات التي سيتم استدعاءها (إجرائية التحقق من صلاحية عملية جداء مثلاً بين تعبيرين رياضيين).

1 - 8 - 1 - مجال تعريف ورؤية المتحولات

نعرف مجال تعريف المتحول بأنه مجموعة أجزاء البرنامج الذي يكون المتحولاً فيها معروفاً وقابلاً للاستخدام وفقاً للمعنى الذي أعطي له عند الإعلان عنه. يختلف معنى هذا المجال من لغة إلى أخرى، ففي لغة كوبول *(COBOL)* تكون التحولات معروفة ومرئية في كافة أجزاء

البرنامج، في حين لا تكون هذه المتحولات معروفة أو مرئية إلا في المقاطع التي يجري فيها الإعلان عن المتحول في لغات برمجة مثل باسكال *(Pascal)* و لغة سي *(C)*.

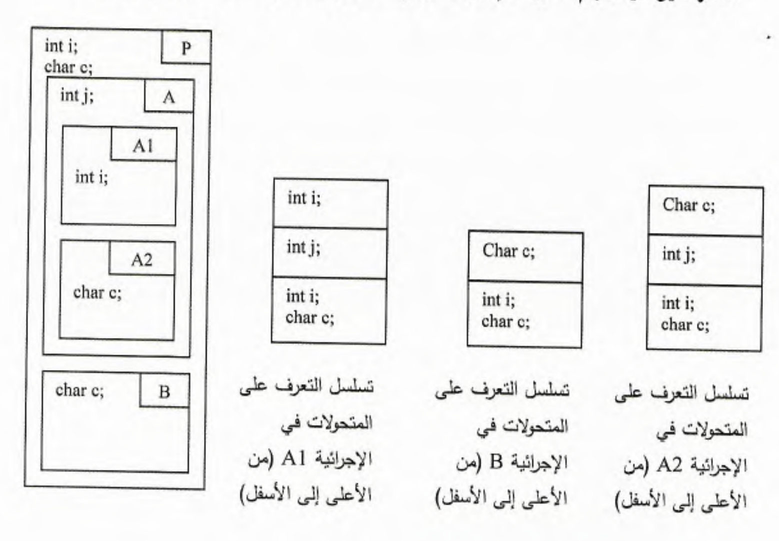
للمساعدة في تحديد مجال تعريف ورؤية المتحول، يستخدم مصمم المترجم بنية معطيات رئيسية هي جدول الرموز *(Symbol Table)*. يكون هدفها تخزين المتحول عند الإعلان عنه مع تحديد بعض العناصر المرتبطة به وخصوصاً نمطه والمقطع الذي ينتمي إليه. لذا يحتاج مصمم هذه البنية إلى وضع بنية تسمح له -عند قيامه بالتحليل الدلالي أثناء التحليل القواعدي، وعند تحليله لتعبير رياضي يستخدم متحول ما- أن يعود إلى جدول الرموز للتأكد من أن هذا المتحول مرئي ومعرَّف في المكان الذي تم استخدامه فيه.

تحتوي بنية رمز في جدول الرموز، على المعلومات التالية:

- اسم المتحول *(X, y, t)*.
- نمط المتحول *(عدد صحيح، عدد حقيقي، ... الخ)*.
- في حال كان الإعلان عن إجرائية فيتم تحديد عدد معاملاتها وأنماط هذه المعاملات.

يمكن ضمن هذا الجدول الاحتفاظ بمعلومة المقطع الذي ينتمي إليه المتحول، أو بناء الجدول نفسه على شكل مجموعة من الجداول التي يرتبط كل منها بمقطع، وتكون الجداول مرتبطة ببعضها البعض تبعاً لهرمية ارتباط المقاطع ببعضها البعض. في الحالة الأخيرة يتم البحث عن متحول *(عند استخدامه)* للتأكد من وجود إعلان عنه وللتأكد من نمطه، في جدول رموز المقطع نفسه، أو في جداول رموز المقاطع التي تحتوي هذا المقطع فقط.

يوضح الشكل التالي هرمية ارتباط مجموعة من جداول رموز ببعضها البعض ضمن برنامج *P* يحتوي على إجرائية *A* وإجرائية *B* معرفتين فيه. وتم تعريف إجرائيتين إضافيتين *A1* و*A2* ضمن الإجرائية *A*.



1 - 8 - 2 - التحقق من الأنماط

عندما تكون اللغة المصدرية منمطة، يجب أن يتحقق المترجم من صحة العمليات التي تتم على هذه الأنماط من حيث ملاءمتها للنمط المعرَّف. يختلف هذا التحقق من لغة برمجة إلى أخرى وتكون قواعد التنميط والقواعد الدلالية الأخرى من مسؤولية مصمم اللغة. فعلى سبيل المثال، لا يمكن في لغة مثل لغة البرمجة *C* جمع عدد حقيقي من النمط *double* مع سلسلة محارف لها النمط *char**. كما لا يمكن تنفيذ عملية جداء عدد صحيح *int* ببنية مركبة *struct*. إلا أن بعض العمليات الأخرى في نفس اللغة، تكون ممكنة. فعلى سبيل المثال،

يمكن إسناد عدد صحيح *int* إلى متحول من النمط الحقيقي *double* أو من النمط *char*.

عموماً، هناك نوعان من التحقق الدلالي: الأول ساكن يتم في مرحلة الترجمة كالأنواع التي ذكرناها سابقاً والآخر ديناميكي يتم عند التنفيذ ولا يمكن للمترجم مراقبته كأن يكون لدينا جدول معرف بعشر خانات *tab[10]* ونستخدم مؤشر *I* فيه القيمة *12* كمؤشر ضمن هذا الجدول أي أن نكتب *tab[i]=10* مع أن *i* تحتوي القيمة *12*.

تتم عملية التحقق من الأنماط عبر حساب أنماط التعابير والمتحولات من خلال إجرائيات يتم استدعاءها أثناء عمليات المسح وضمن القواعد الصرفية. فعلى سبيل المثال يمكننا أن نكتب ضمن قاعدة صرفية توصف عملية الإسناد ما يلي:

```
I → Id=E
{
    if (Id.Compute_type() == E.Compute_type())
        return true;
    else
        return Error("Incompatible type", LineNb);
}
```

حيث نلاحظ مما سبق أننا نحتاج لمعرفة نمط كل من *I* و*E* قبل تنفيذ الإجراءات السابقة.

كما يمكننا أن نكتب ضمن قاعدة صرفية خاصة بعملية جمع تعبيرين رياضيين مايلي:

```
E → E + E
{
    if ((E(1).Compute_type() = = integer) && (E(2).Compute_type() = = integer))
        E(0).SetType(integer)
}
```

يمكننا تعريف قواعد التحقق من الأنماط بلغة محكية أو استخدام توصيف رياضي لها يساعدنا في توضيحها (كما هو الحال في القواعد الصرفية) وبحيث يسهل تحويلها إلى إجرائيات وإدراجها ضمن القواعد الصرفية عند الحاجة. لتنفيذ ذلك، لنعرف التدوينين التاليين:

$$\alpha \Rightarrow e : \tau$$

"ضمن السياق *a*، يمثل *e* تعبيراً منمطاً، ويكون من النمط τ"

$$\alpha \Rightarrow_L e : \tau$$

"ضمن السياق *a*، يمثل *e* *(Left-value)* تتقبل الإسناد، وتكون من النمط τ"

ويمكن اعتباراً من التدوينين السابقين تصميم قواعد التنميط التالية وفق آليات الاستنتاج الموضحة فيما يلي حيث يؤدي تحقق الشروط الموجودة في أعلى كل خط إلى تحقق ما هو وارد في أسفل كل خط. تساعد هذه القواعد عند تصميمها ووضعها بشكل واضح، في تنفيذ

إجرائيات التحليل الدلالي بشكل أبسط عند برمجة المترجم. فيما يلي بعض الأمثلة عن قواعد حساب أنماط التعابير الرياضية:

Rule	Description
$\dfrac{\alpha \Rightarrow_L e : int}{\alpha \Rightarrow ++e : int}$	ضمن السياق α، إذا كانت *e* عبارة عن *(Left-value)* تتقبل الإسناد ومن النمط *int*، فإن *e++* مقبولة ضمن السياق نفسه وتكون من النمط *int*.
$\dfrac{\alpha \Rightarrow_L e : int}{\alpha \Rightarrow ++e : int}$	ضمن السياق α، إذا كانت *e* عبارة عن *(Left-value)* تتقبل الإسناد ومن النمط *int*، فإن ++e مقبولة ضمن السياق نفسه وتكون من النمط *int*.
$\dfrac{\begin{array}{l}\alpha \Rightarrow e1 : \tau 1 \\ \alpha \Rightarrow e2 : \tau 2 \\ \tau 1, \tau 2 \in \{int, double, float\} \\ op \in \{<, <=, >, >=\}\end{array}}{\alpha \Rightarrow e1\ \ op\ \ e2 : boolean}$	ضمن السياق α، إذا كان *e1* عبارة عن تعبير من النمط τ*1* وكان *e2* عبارة عن تعبير من النمط τ2 وكانت كلاً من τ*1* و τ*2* هي إما *int* أو *double*، أو *float* فإن تطبيق عملية مقارنة *op* على التعبيرين، يعطينا تعبيراً من النمط *boolean* ضمن السياق α.

1 - 9 - توليد الرماز

1 - 9 - 1 - البنية الوسيطة

عند بناء مترجم، يتم استخدام بنية وسيطة لتمثيل البرنامج المصدري الذي تتم ترجمته. تكون هذه البنية الوسيطة على شكل شجرة ندعوها شجرة القواعد المجردة، لأنها تمثل القواعد الصرفية ولكن بصيغة بنية معطيات شجرية. عند انتهاء مسح القواعد الصرفية تكون عملية التحليل المفرداتي والقواعدي والدلالي قد انتهت، وبالإضافة لما سبق، تكون عملية بناء هذه البنية الوسيطة قد انتهت بحيث يمكن اعتباراً منها توليد الرماز عبر عبور الشجرة الممثلة للبرنامج المصدري وتوليد الرماز المناسب.

يمكن لبنية عقدة من الشجرة أن تكون على الشكل التالي:

```
/*      types of objects                              */
typedef enum tempObjType {
  DECL_O,        /* declaration                       */
  STMT_O,        /* statement (no return)             */
  EXP_O,         /* expression (return)               */
  LIT_O,         /* literal (constant)                */
  ID_O           /* identifier                        */
} ObjType;
```

حيث يمثل كل جزء من أجزاء هذه البنية إحدى إشكال العقد التي يمكن أن تكون عليها هذه الشجرة. إذ يمكن للعقد أن تمثل عملية تعريف متحولات، أو جملة تحتوي تعليمة شرطية أو تعليمة حلقية. كما يمكن أن تعبر العقدة عن تعبير رياضي أو منطقي، أو متحولات

وعمليات إسناد خاصة بها. فعلى سيبل المثال يمكن لبرنامج أن يمثل بالشكل التالي:

```
x := a + b;
y := a + b;
while (y > a) {
        a := a + 1;
        x := a + b
}
```

Program
:=
...
while
x
+
a
b
>
y
a
Block
:=
...
a
+
a
1

1 - 9 - 2 - تنظيم الذاكرة وتنفيذ عملية الحساب

قبل البدء بتوليد الرماز، يجب طرح مجموعة من الأسئلة عن كيفية تنظيم الذاكرة:

- عند إرجاع الإجرائيات لقيم، أو عند استخدام تعليمات القفز من تعليمة مثل تعليمة *break* في لغة *C*.
- عند تخزين المتحولات.
- عند الحجز الديناميكي للذاكرة.

هناك بعض الأسئلة التي يتوجب طرحها والتي تكون مرتبطة ارتباط مباشر باللغة المصدرية التي نترجمها:

- هل يمكن أن تكون الإجرائيات، عودية؟
- هل يمكن لإجرائية أن تستخدم متحولات غير محلية وغير معرفة ضمن مقطعها؟
- كيف يتم تمرير المعاملات عند استدعاء الإجرائية؟

- هل يجب أن تكون عملية تحرير مساحة الذاكرة التي يتم حجزها ديناميكياً، عملية يدوية يقوم بها المبرمج الذي يبرمج باللغة المصدرية باستخدام تعليمات خاصة، أم عملية آلية لا تحتاج لتدخل المبرمج؟
- ما مصير المتحولات المحلية المعرفة ضمن مقطع إجرائية، عند الانتهاء من استدعاء الإجرائية؟

كما توجد عدة أسئلة أخرى تتعلق بالنظام واللغة التي تتم عملية الترجمة باتجاهها:

- ما هو طول عنوان عند حجز الذاكرة؟
- ما هي الكيانات التي يمكن عنونتها مباشرةً في الذاكرة؟
- ما هي تعليمات الوصول إلى كيانات ومقاطع من الذاكرة قابلة للعنونة المباشرة؟
- ما هي ضوابط عملية العنونة؟

تؤثر الإجابات على الأسئلة السابقة في عملية تنظيم الذاكرة. عموماً، يقسم نظام التشغيل الذاكرة المخصصة للبرنامج التنفيذي إلى 4 مناطق أساسية:

الرماز المولد
معطيات ساكنة
مكدس التحكم
منطقة توسع
المكوم

- **منطقة المعطيات الساكنة:** يكون حجم بعض المعطيات معروفاً منذ مرحلة الترجمة، لذ، وعند توليد الرماز، يتم كتابة تعليمات لحجز أماكن لهم وتخزينهم في أماكن محددة سلفاً من الذاكرة وهي الأماكن المخصصة لتخزين المعطيات التي ندعوها ساكنة *(Static)*، مما يساعد على الانتهاء من توليد تعليمات حجز أماكنهم منذ مرحلة الترجمة وأثناء توليد الرماز.

- **مكدس التحكم:** وهو مكدس يسمح بإدارة عمليات استدعاء الإجرائيات وعمليات الإرجاع التي تقوم بها. إذ تمتلك جميع الآلات، آليات خاصة تسمح بكتابة رماز خاص عند استدعاء إجرائية، يقوم هذا الرماز بحجز سجل خاص لحفظ حالة البرنامج الذي قام بالاستدعاء ضمن هذه المنطقة، وإعادته إلى حالته عند انتهاء الاستدعاء.

- **المكوم:** وهو المكان الذي يجري فيه حجز المتحولات المعرفة كمتحولات ديناميكية، بحيث يتم تعريف حجمها أثناء التنفيذ ضمن هذه المنطقة وضمن منطقة التوسع. وتتم إدارة المكوم من قبل الآلة *(نظام التشغيل)* وفقاً لآليات خاصة تعتمد إما على تنفيذ عمليات تحرير آلي للمساحات المحجوزة عند انتهاء صلاحية المتحولات الخاصة بهذه المساحات، أو على انتظار تعليمات تحرير خاصة يجب أن يتضمنها الرماز المكتوب. كما تقوم الآلة بعملية إدارة المساحة المخصصة للتكويم وفق آليات تنظيم تسمح بأمثلة عملية حجز المساحات فيها والحفاظ دائماً على مساحات كبيرة حرة قابلة للحجز.

بالإضافة لكل ما سبق، نحتاج لإدارة عملية حساب قيم تعابير رياضية ومنطقية، إلى استثمار لغة الخرج لتخزين القيم المرحلية للتعابير قبل الوصول لحساب قيمتها النهائية، ما يعني أننا بحاجة

لاستثمار إمكانيات التكديس والتخزين والاسترجاع في لغة الخرج بحيث يتم بناء الخرج على نحو يحقق عمليات حساب صحيحة. من أهم آليات الحساب المتبعة والتي تصنف الآلات من خلالها:

- آلية الحساب بالتكديس في الآلة ذات المكدس *(Stack Machine)*.
- آلية الحساب بالتخزين في الآلة ذات السجلات *(Register Machine)*.

بشكل عام يمكن إيضاح الفرق بين الآلتين في المثالين التاليين وبحيث سنستعرض بالتفصيل بنية آلة ذات مكدس في الفصل الأخير عند الكلام عن الآلة الافتراضية التي سنستخدمها في مشروع المترجم.

يكون رماز خاص بعملية جمع $x+y$ بآلة ذات سجلات على النحو التالي:

STORE Ri x	→ *Let the value of register i be stored at address x*
LOAD Ri x	→ *Fetch the value of x, place it in register i*
STORE Rj y	→ *Let the value of register j be stored at address y*
LOAD Rj y	→ *Fetch the value of y, place it in register j*
ADD Ri Rj	→ *Fetch the value of register j, add it to the value in register i*

يكون رماز خاص بعملية جمع $x+y$ بآلة ذات مكدس على النحو التالي:

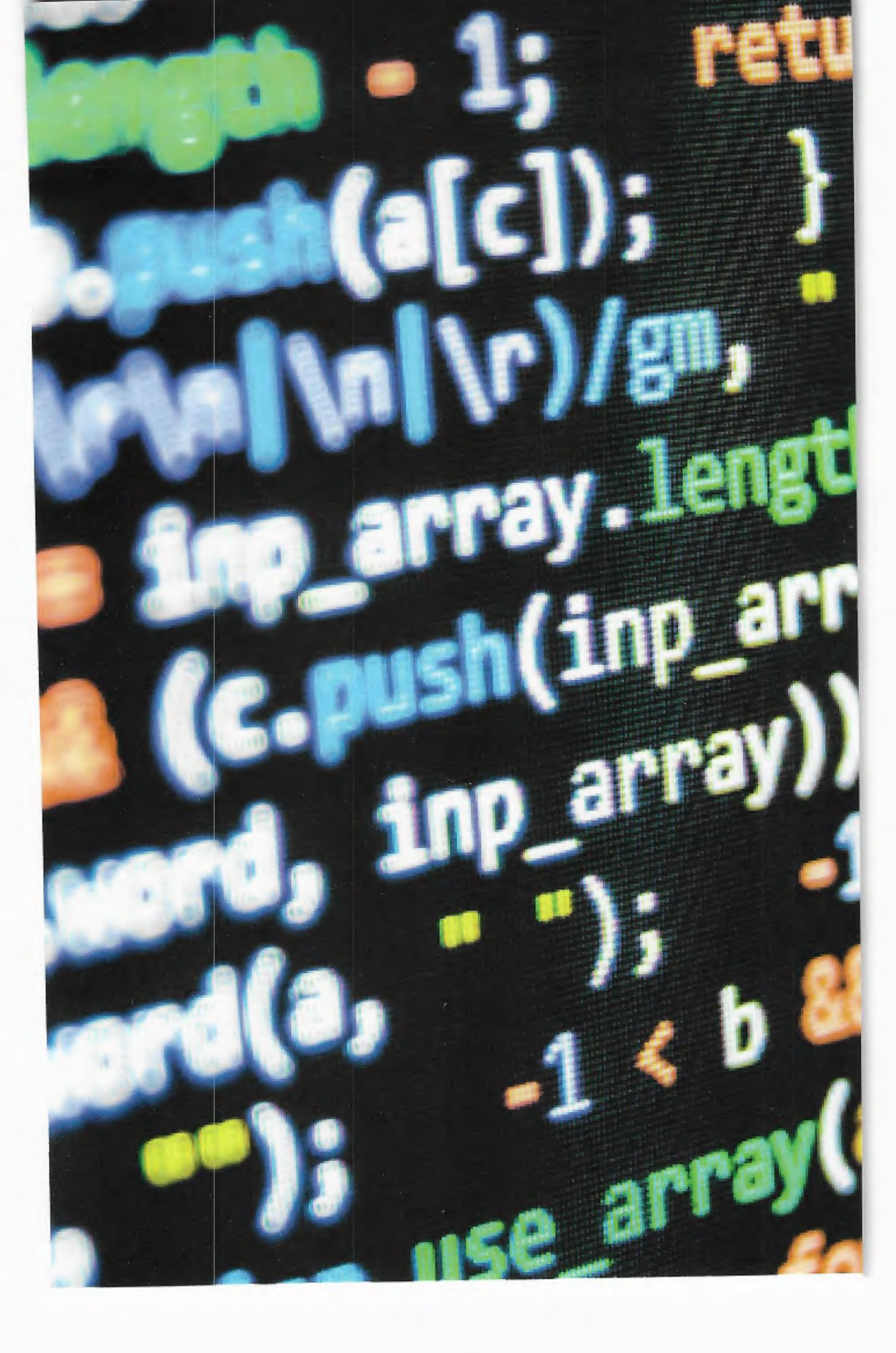
STORE x	→ *Store the value of x at address x*
STORE y	→ *Store the value of y at address y*
LOAD x	→ *Fetch the value from address x, push it on to the stack*
LOAD y	→ *Fetch the value from address y, push it on to the stack*
ADD	→ *Replace the top two values on the stack by their sum*

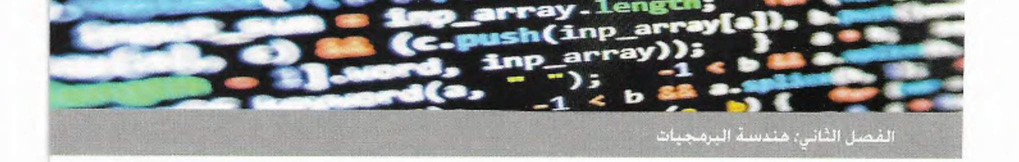
1 - 9 - 3 - توليد الرماز المقابل للتعليمات

يمكن للجدول التالي أن يوضح لنا، الرماز الواجب توليده بلغة آلة أمام كل نوع من أنواع التعليمات البرمجية الأساسية:

التعليمة البرمجية بلغة الآلة	التعليمة البرمجية باللغة المصدرية
execute(C1); execute(C2)	*execute(C1;C2)*
evaluate E *JUMPIF(0) g* *execute C1* *JUMP h* *g: execute C2* *h:*	*execute(if E then C1 else C2)*
JUMP h *g: execute C* *h: evaluate E* *JUMPIF(1) g*	*execute(while E do C)*
g: evaluate E *JUMPFALSE h* *execute C* *JUMP g* *h:*	
g: execute C *evaluate E* *JUMPIF(1) g*	*execute(repeat C until E)*
evaluate E *assign I*	*execute(I := E)*

التعليمة البرمجية بلغة الآلة	التعليمة البرمجية باللغة المصدرية
pass-args A *CALL p* */* p = address of routine L */*	*execute(L (A))*
evaluate(E) *fetch_Address(V)*	*pass-args(E)* *pass-args(@V)*
address(I)	*fetch(I)*
STORE address(I)	*assign(I)*
evaluate(E1) *evaluate(E2)* *op*	*evaluate(E1 op E2)*


1;
(a[c]); }
\r)/gm,
inp_array.
(c.push(inp_
inp_array))
" ");
-1 < b
");
use array(

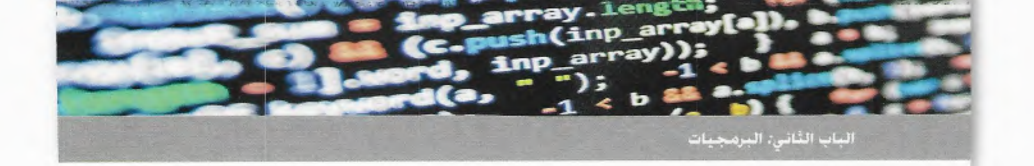


الفصل الثاني

هندسة البرمجيات

(Software Engineering)

- مفاهيم أساسية في هندسة البرمجيات
- هندسة المتطلبات
- نمذجة النظام
- تصميم النظام
- التنجيز والتسليم والاختبار



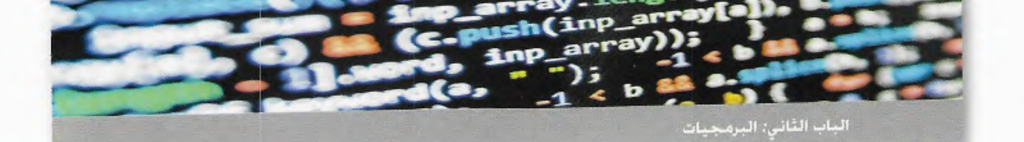
2 - 1 - مفاهيم أساسية في هندسة البرمجيات

Software Engineering Basics

2 - 1 - 1 - المقدمة العامة

يهدف هذا الفصل إلى التعريف بأساسيات علم هندسة البرمجيات في تطوير المنتجات البرمجية مع استعراض لأهم الإجرائيات والطرق والأدوات المستخدمة. يستعرض الفصل تعريف هندسة البرمجيات متناولاً الأطوار الأساسية في دورة حياة التطوير البرمجي *System Development Life Cycle (SDLC)* بأشكالها المتعددة من النموذج الشلالي الى الحلزوني تحت بند النماذج ذات التخطيط المسبق ومن ثم ينتقل إلى التعريف عن أهم إجرائيات التطوير الرشيق.

بعد استعراض أهم نماذج التطوير البرمجي المستخدمة يتم الخوض في تفاصيل أطوار التطوير البرمجي بدءاً من لحظة طلب النظام *System request* ومن ثم عرض إجرائيات الدراسة التحليلة بما يتضمنه ذلك من هندسة لمتطلبات النظام البرمجي بنوعيها الوظيفية وغير الوظيفية. مناقشة الدراسة التصميمية يأتي تالياً ومنها يتم استعراض آليات ترجمة التصاميم إلى أكواد برمجية في مرحلة التنجيز يليها مرحلة النشر حيث يتم مناقشة آليات تسليم المنتج البرمجي الى الزبون بما يتخلله ذلك من عمليات اختبار مختلفة قبل وبعد التسليم.



2 - 1 - 2 - تعريف هندسة البرمجيات

هندسة البرمجيات هو العلم الذي يهتم باستخدام الانضباط الهندسي في تطوير المنتجات البرمجية عن طريق توصيف مجموعة من الأطر الإجرائية *Process framework* والطرق *Methods* والأدوات *tools* في جميع جوانب إنتاج البرمجيات بدءاً من المراحل الأولى من توصيف النظام الى الحصول على التطبيق بما يحقق المعايير الاقتصادية ومواصفات الجودة *Quality* حسب ما يبينه الشكل *(1)*.



الشكل *(1)*: مفهوم هندسة البرمجيات

2 - 1 - 3 - إطار عمل الإجرائية البرمجية *Software Process Framework*

يمثل الشكل *(2)* اطار عمل الإجرائية البرمجية الذي يتألف من مجموعة من الأنشطة *activities* ضمن إطار عمل قابل للتطبيق لجميع المشاريع البرمجية بغض النظر عن حجمها أو تعقيدها ويمكن تقسيمها الى صنفين:

1 - النموذج الإجرائي *Process Model*

وهو مجموعة من الأنشطة *work tasks* التي تهتم بإنتاج التطبيقات البرمجية عن طريق تقسيمها الى مجموعة من المراحل *milestones* بحيث تهتم كل مرحلة بإنتاج جزء قابل للقياس *work products*

والتحقق منه عن طريق تطبيق مجموعة اجرائيات ضبط الجودة *quality checks*. ترتبط هذه المرا حل فيما بينها حسب قواعد تدفق العمل *work flows*.

Process framework

Framework activities

work tasks
work products
milestones & deliverables
QA checkpoints

Umbrella Activities

الشكل *(2)*: إطار عمل الإجرائية البرمجية

يقسم النموذج الاجرائي لتطوير البرمجيات الى صنفين أساسيين:

- **نماذج الاجرائيات التي تعتمد على التخطيط** *Models Plan-driven Process*: حيث يتم التخطيط لجميع الأنشطة العملية مسبقا ويتم قياس التقدم مقابل هذه الخطة.
- **الإجرائيات الرشيقة** *Agile Processes*: هنا يتم التخطيط بشكل تدريجي وسهل بحيث تمكن من تغيير العملية لتعكس تغيير متطلبات العملاء.

2 - نشاطات المظلة *Umbrella Activities*

مجموعة من الأنشطة الشاملة يتم تطبيقها خلال مراحل تطوير البرمجيات لمساعدة فريق البرمجيات في إدارة ومراقبة تقدم المشروع البرمجي، والتحقق من الجودة، وتتبع المخاطر. تشمل هذه الأنشطة بشكل عام مايلي:

- **تتبع والتحكم بسير المشاريع البرمجية *Software project tracking and control*:** يسمح لفريق العمل بتقييم التقدم المحرز في تنفيذ خطة المشروع واتخاذ أي إجراء ضروري للحفاظ على الجدول الزمني.

- **إدارة المخاطر *Risk management*:** تقييم المخاطر التي قد تؤثر على نتائج المشروع أو جودة المنتج.

- **إدارة جودة البرمجيات *Quality management*:** تحديد والقيام بالأنشطة المطلوبة لضمان جودة البرمجيات.

- **المراجعات الفنية *Technical reviews*:** تقييم منتجات هندسة البرمجيات في محاولة للكشف عن الأخطاء وإزالتها قبل نشرها إلى النشاط التالي.

- **القياسات البرمجية *Software metrics*:** تحديد وتجميع البيانات حول الإجرائيات والمشاريع والمنتجات التي تساعد الفريق في تحسين البرامج التي تلبي احتياجات أصحاب المصلحة *Stakeholders*.

- **إدارة تكوين البرمجيات *Software configuration management*:** تحديد والتعامل مع آثار التغيير الممكن حدوثه أثناء عملية التطوير البرمجي.

- **إدارة إعادة الاستخدام *Reusability management*:** تحديد معايير إعادة استخدام منتجات العمل (*بما في ذلك مكونات البرامج*) ووضع آليات لتحقيق مكونات قابلة لإعادة الاستخدام.

- **إعداد وإنتاج المنتج *Work product preparation and production*:** يشمل الأنشطة المطلوبة لخلق منتجات العمل مثل النماذج والوثائق والسجلات والنماذج والقوائم.

2 - 1 - 4 - نماذج الإجرائيات المعتمدة على التخطيط
Plan-driven process models

يتألف النموذج الإجرائي العام *Generic Process Model* من خمسة مراحل أساسية:

- **الاتصالات *Communication*:** تتضمن مهمة التواصل والتعاون مع العملاء وأصحاب المصلحة *(Stakeholders)* لجمع المتطلبات التي تساعد على تحديد ميزات البرنامج ومهامه الرئيسية.
- **التخطيط *Planning*:** يتضمن إعداد "خريطة" تساعد على توجيه الفريق البرمجي من خلال وصف المهام التقنية التي يتعين القيام بها، والمخاطر التي من المرجح ان يتم اعتراضها، والموارد التي ستكون مطلوبة، ومنتجات العمل التي سيتم إنتاجها، والجداول الزمنية التي يجب السير وفقها.
- **النمذجة *Modelling*:** تتضمن خلق نماذج تحليلية من أجل فهم متطلبات البرامج وتصميمية من أجل بناء المنتج البرمجي الذي يلبي هذه المتطلبات.
- **أعمال البناء *Construction*:** تتضمن توليد الرماز *code* والاختبار المطلوب للكشف عن الأخطاء في التعليمات البرمجية.
- **النشر *Deployment*:** تسليم المنتج البرمجي ككيان كامل أو جزئي إلى الزبون وتوفير التغذية الراجعة لجمع آراء الزبائن لتقييم المنتج.

تم تطوير عدد من النماذج الإجرائية لتطوير المنتجات البرمجية والتي تستوعب بطبيعتها أنشطة النموذج الإجرائي العام ولكن تختلف كل منها من حيث مجال الاستخدام وتدفق الاجرائيات والميزات والإيجابيات.

1 - نموذج الشلال *Waterfall model*

يوضح الشكل *(3)* نموذج الشلال لتطوير المنتجات البرمجية حيث يتألف من الأطوار الرئيسية المتواجدة في النموذج الإجرائي العام والمترابطة بشكل تسلسلي باتجاه واحد.

- **الاستخدام:** يتم تفضيل هذا النموذج في الحالات التالية:

- المتطلبات واضحة ومفهومة.
- المتطلبات موثقة بشكل جيد وثابتة (غير قابلة للتغيير).
- التقنيات مفهومة وغير ديناميكية.
- التغييرات محدودة في إجرائيات التطوير.
- زمن تنفيذ المشروع متوسط لقصير نسبياً.
- خبرة الفريق البرمجي ضعيفة إلى متوسطة.
- يصلح مهما كان حجم المشروع.

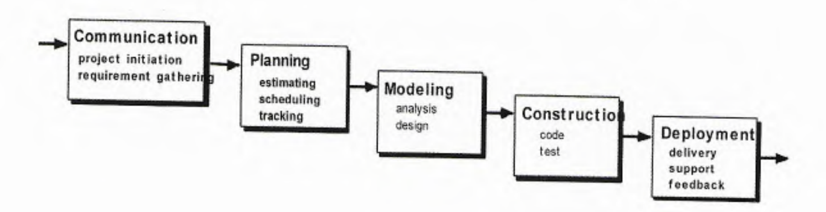
- **الميزات:**

- تسلل الإجرائيات خطي.
- درجة التحكم في إجرائية التطوير البرمجية عالية بحكم وجود المراحل ذات المهام المحددة والمعرفة مسبقاً.
- درجة التوثيق عالية.
- تقييم المنتج البرمجي في نهاية عملية التطوير وعند التسليم.

- **السلبيات:**

- تسلل الإجرائيات ذو اتجاه وحيد.

- الصعوبة في مواكبة التغيير المستمر في متطلبات المشروع.
- هدر للوقت واليد العاملة بحكم أن كل طور لا يبدأ إلا بعد انتهاء الطور السابق.
- تسليم المشروع لا يتم إلا بعد انتهاء كامل إجرائيات التطوير البرمجية.



الشكل *(3)*: نموذج الشلال

2 - النموذج التزايدي *Incremental model*

يوضح الشكل *(4)* النموذج التزايدي الذي يقسم المشروع البرمجي إلى عدد من الإصدارات بحيث يقوم كل إصدار بتلبية جزء من المتطلبات الوظيفية للمشروع البرمجي.

- **الاستخدام:** يتم تفضيل هذا النموذج في الحالات التالية:

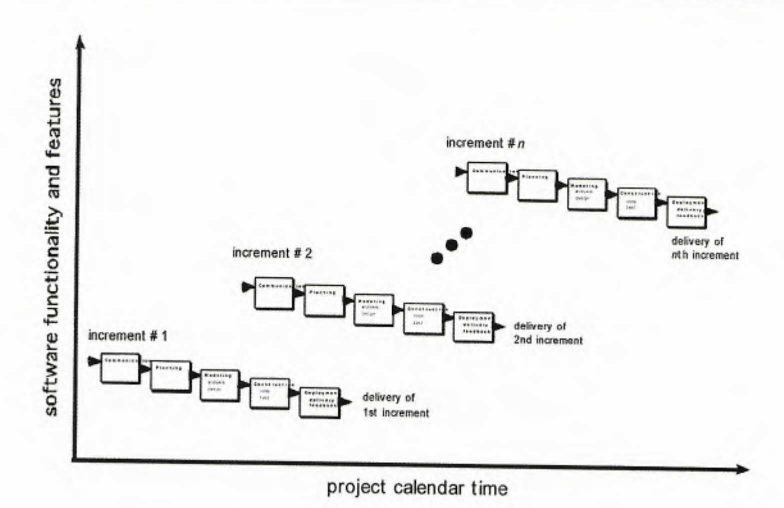
- المتطلبات متغيرة خلال إجرائيات التطوير البرمجي.
- إصدار المنتج البرمجي مطلوب بسرعة.
- الفريق البرمجي غير متاح على كامل فترة المشروع البرمجي.

- **الميزات:**

- تناقص تكلفة تغيير المتطلبات البرمجية.
- الحصول على رأي الزبون على الإصدار الذي تم تسليمه وبالتالي إمكانية التقييم والتعديل أثناء إعداد الإصدار اللاحق.
- سرعة تسليم إصدارات برمجية متعاقبة إلى الزبون وبالتالي البدء بالاستفادة من المنتج البرمجي بسرعة أكبر من نموذج الشلال.
- إمكانية العمل على أكثر من إصدار بشكل متوازي.

- **السلبيات:**

- صعوبة قياس التقدم في إنجاز المشروع البرمجي.
- زيادة تكلفة ومدة المشروع وخصوصاً في حال توثيق كامل الإصدارات.
- ترهل البنية التركيبية للمنتج البرمجي بزيادة عدد الإصدارات.
- زيادة التكلفة بسبب الحاجة إلى إعادة تركيب وترتيب بنية المنتج البرمجي باستمرار حتى تصبح من الصعوبة القيام بإجراء التغييرات والتحسينات.



الشكل *(4)*: النموذج التزايدي

3 - النموذج الحلزوني *Spiral model*

يوضح الشكل *(5)* النموذج الحلزوني الذي يمتاز بدراسة وتحليل المخاطر التي قد تؤثر على سير المشروع البرمجي وطبيعته التزايدية التكرارية.

- **الاستخدام:** يتم تفضيل هذا النموذج في الحالات التالية:

- حجم المشروع البرمجي ضخم وأكثر عرضة للفشل.
- هناك ضرورة لتقييم المخاطر والتكاليف.
- المشروع البرمجي متوسط إلى عالي الخطورة.
- زمن المشروع طويل.
- الزبون غير واثق من احتياجاته.
- المتطلبات معقدة.

- المنتج البرمجي فريد من نوعه.
- تغييرات كبيرة في المتطلبات.

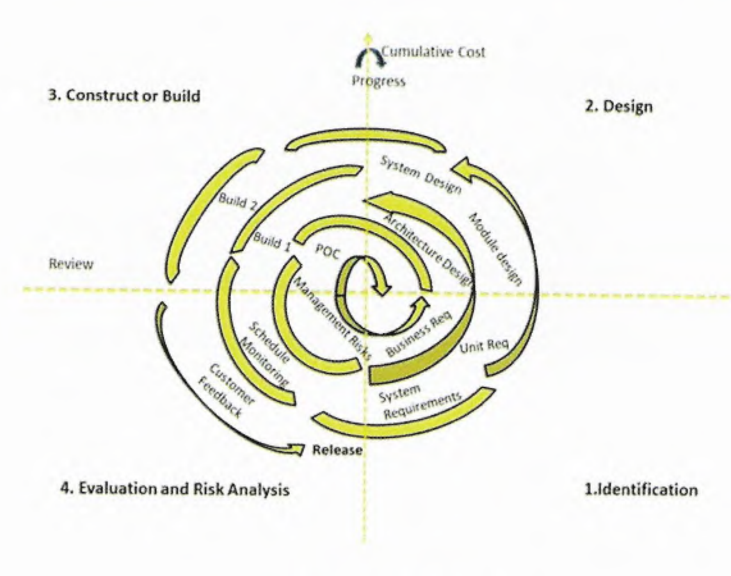
- الميزات:

- زيادة فرصة تجنب المخاطر.
- درجة تحكم وتنظيم وتوثيق عالية.
- تقبل تعديل أو إضافة في المتطلبات البرمجية.
- إمكانية تسليم الإصدار الأول في وقت مبكر.
- درجة الوثوقية عالية.

- السلبيات:

- مكلف جداً.
- يحتاج إلى فريق خبير لدراسة وتحليل المخاطر.
- لا يناسب المشاريع الصغيرة.
- بحاجة إلى فريق برمجي كبير ومتمرس.
- قد يتطلب تحضير نموذج أولي *prototype* من المنتج البرمجي أثناء عملية التطوير.





الشكل *(5)*: النموذج الحلزوني

4 - النموذج الإجرائي الموحد *Unified Process model*

يوضح الشكل *(6)* النموذج الإجرائي الموحد الذي يعتمد على لغة النمذجة الموحدة *UML* في بناء تطبيقات غرضية التوجه ويمتاز بدراسة وتحليل المخاطر التي قد تؤثر على سير المشروع البرمجي وطبيعته التزايدية التكرارية لتسليم منتج برمجي متكامل ذو جودة عالية تلبي كامل احتياجات الزبون ضمن الوقت والتكلفة المحددة.

يتألف النموذج من عدة أطوار

(Transition – Construction - Elaboration – Inception) Phases

حيث يتضمن كل طور جميع مراحل النموذج الإجرائي العام (تخطيط *Planning* – تحليل *Analysis* – تصميم *Design* – تنجيز *Implementation* – اختبار *Tasting* – دعم فني *Support*).

يتألف التكرار الواحد *Iteration* من عدة مراحل ويتألف الطور الواحد من عدة تكرارات والتي يعتمد عددها على حجم المشروع. يميز كل طور إنتاج منتج مرحلي محدد *Work Product*.

- **الاستخدام:** يتم تفضيل هذا النموذج في الحالات التالية:

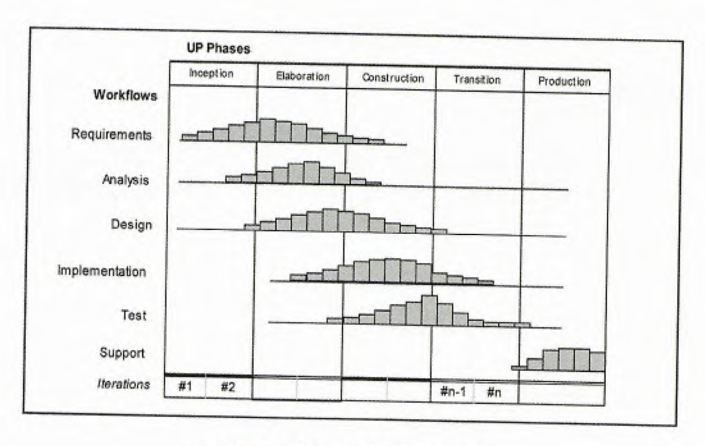
- الشركة البرمجية ضخمة وتعمل بمشاريع متعددة.
- المشاريع البرمجية غرضية التوجه *Object oriented*.
- أصحاب العلاقة *Stakeholders* نشيطين وقريبين من المشروع.

- **الميزات:**

- الحصول على آراء المستخدمين بشكل مستمر.
- تحقيق التعاون المستمر بين أعضاء الفريق البرمجي.
- تسليم إصدارات برمجية متعددة بشكل سريع ومنتظم.
- استخدام أدوات النمذجة المعروفة.
- إعادة استخدام المكونات البرمجية لمشاريع سابقة.
- التركيز على الاستمرارية والجودة.
- اكتشاف المشاكل في مراحل مبكرة من المشروع.
- الاستخدام الأمثل للموارد.
- تطوير أدوات تحليل وإدارة الخطورة.

- **السلبيات:**

- النموذج عالي التعقيد وبالتالي يمكن أن يعاني من سوء التطبيق وفقدان التحكم.
- يحتاج إلى فريق برمجي كبير وخبير.



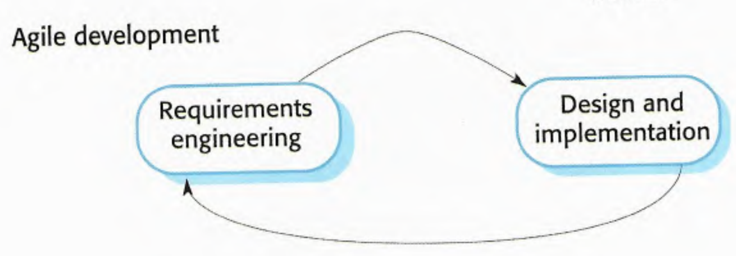
الشكل *(6)*: النموذج الإجرائي الموحد

2 - 1 - 5 - الإجرائيات الرشيقة *Agile Processes*

تهدف الإجرائيات الرشيقة إلى إنقاص زمن تطوير المنتجات البرمجية بشكل كبير بحيث تعتمد على الربط والدمج بين أطوار التطوير المختلفة من تحديد المواصفات الى التصميم والتنجيز لإنتاج اصدارات تزايدية من المنتج المراد تطويره بتكرارات متعددة كما هو موضح في الشكل *(7)*. يعتمد التطوير الرشيق للمنتجات البرمجية على توافر المبادئ التالية:

- القدرة على الاستجابة السريعة والمتكيفة للمتغيرات.
- تأمين تواصل فعال بين كل أصحاب المصلحة *Stakeholders*.
- ضمان وجود الزبون ضمن أعضاء الفريق البرمجي.
- وجود فريق برمجي خبير ومتفاهم.
- تطوير المنتج البرمجي على عدة إصدارات تزايدية.
- ضمان البساطة سواء في آلية تطوير المنتج البرمجي أو في المنتج

نفسه.



الشكل (7): التطوير الرشيق

وبما أن الإجرائيات الرشيقة تعتمد بشكل أساسي على خبرة ومهارة الفريق البرمجي أكثر من اعتمادها على وجود إجرائيات واضحة المعالم، وجب أن يتمتع هذا الفريق بالمواصفات التالية:

- الكفاءة.
- التركيز.
- التعاون.
- القدرة على اتخاذ القرار.
- القدرة على حل المشاكل المختلفة.
- الثقة والاحترام المتبادل.
- التنظيم الذاتي.

وفيما يلي نبين متى ينصح استخدام الإجرائيات الرشيقة وميزاتها وسلبياتها:

- **الاستخدام:** يتم تفضيل هذه النماذج في الحالات التالية:

- المنتج البرمجي صغير إلى متوسط الحجم.
- إمكانية التزام الزبون في إجرائيات تطوير المنتج البرمجي.
- تطوير المنتجات البرمجية الجديدة أكثر من حالات الصيانة البرمجية.

- **الميزات:**

- تخفيض التكاليف.
- إمكانية التعديل بشكل سريع وفعال في أي مرحلة من مراحل تطوير المنتج البرمجي.
- التسليم السريع للإصدارات المتعددة.

- **السلبيات:**

- عدم وجود إجرائيات رسمية ضابطة للأداء يؤدي في بعض الأحيان إلى خلق مشاكل وخصوصاً في تحديد التزامات الفرق والعقود وخصوصاً في الشركات الضخمة.
- عدم وجود وثائق تحليلية وتصميمية مما يزيد في صعوبة الصيانة والتطوير.
- ضعف إجرائيات ضبط الجودة بالمقارنة مع النماذج ذات التخطيط المسبق.

تم تطوير عدد من النماذج الرشيقة لتطوير المنتجات البرمجية والتي تشترك فيما بينها المبادئ نفسها سابقة الذكر ولكن تختلف كل منها من حيث مجال الاستخدام وتدفق الاجرائيات والميزات والايجابيات.

1 - نموذج البرمجة القصوى (XP) *Extreme programming*

يوضح الشكل *(8)* نموذج البرمجة القصوى الذي يعتمد بشكل أساسي في تطوير المنتجات البرمجية على التطوير التكراري لعدة إصدارات بحجوم صغيرة بحيث لا يتجاوز تسليم كل إصدار أكثر من أسبوعين. يتألف نموذج البرمجة القصوى بشكل رئيسي من الأطوار التالية:

- التخطيط *Planning*

- يعتمد على التخطيط البسيط للموارد والتكاليف.
- يبدأ بإيجاد قصص المستخدم *User stories*.
- تقسيم قصص المستخدم إلى عدد من إجرائيات التنفيذ التي يمكن توزيعها على أعضاء الفريق البرمجي.
- تحليل كل سيناريو *(قصة مستخدم)* بالتعاون الوثيق مع الزبون لتحديد الأولوية والتكلفة.
- تحديد مجموعة القصص التي يجب أن يتضمنها كل إصدار من المنتج البرمجي مع الالتزام بتاريخ التسليم.

- التصميم *Design*

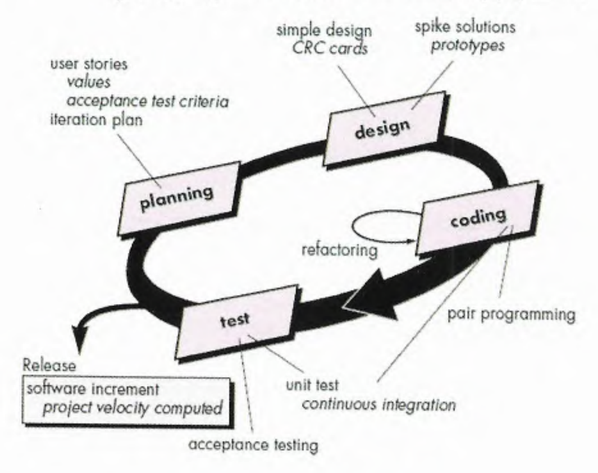
- يتبع قاعدة البساطة *Keep It Simple (KIS)*.
- يستخدم بطاقات التصميم *(class responsibility collaborator (CRC)*.
- يعتمد على إيجاد نماذج تصميمية أولية *Prototype* من أجل حل المشاكل الحدية.

- البرمجة *Coding*

- تعتمد على مبدأ البرمجة الثنائية *Programing Pair* حيث يتولى برمجة كل مهمة مبرمجان يعملان سوياً أحدهما يقوم بأعمال البرمجة والآخر يثوم بالمراجعة ثم يتبادلان الأدوار مما يضمن الحصول على برنامج بأخطاء بسيطة.
- تدعم مبدأ إعادة الهيكلة بشكل دوري *Refactoring* لضمان جودة وانسجام الكود البرمجي.
- يدعم مبدأ اختبار الواحدة *Unit testing* على كل جزء برمجي يتم تطويره من قبل نفس المبرمجين.

- الاختبار *Testing*

- يعتمد على الاختبارات التزايدة المكثفة أثناء التطوير وما قبل التسليم وما بعده لضمان جودة المنتج البرمجي.

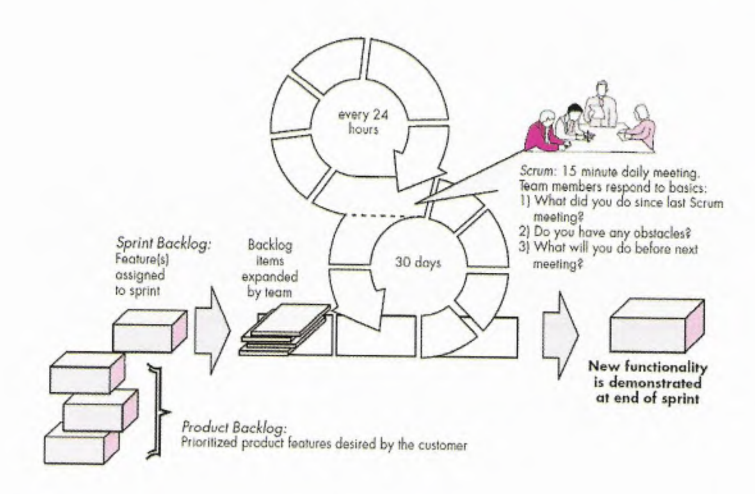


الشكل *(8)*: نموذج البرمجة القصوى

2 - نموذج سكرم *Scrum*

يوضح الشكل *(9)* نموذج سكرم *Scrum* الذي يعتبر من أهم النماذج الرشيقة ويتألف من المراحل التالية:

- تحديد الأهداف الأساسية للمشروع.
- وضع البنية التصميمية المعمارية للمشروع.
- تحديد المتطلبات بشكل تراكمي *Project backlog*.
- تقسيم المتطلبات إلى عدد من التزايدات السريعة *Sprints* بحيث لا تتجاوز دورة *Sprint* أكثر من شهر.
- وضع خطة لكل *Sprint* على حدى.
- اختيار فريق العمل لإنجاز المهمات الموجودة في كل *Sprint* بحيث لا يتجاوز عددهم *7* أشخاص.
- اختيار ممثل عن الزبون يطلق عليه *Project owner*.
- اختيار *Scrum master* الذي يعتبر عضواً أساسياً في فريق العمل والذي يقوم بالإضافة إلى عمله بتنسيق الاجتماعات اليومية لفريق العمل *(اجتماع يومي لمدة 15 دقيقة)* و التأكد من سير العمل بشكل صحيح وفق المتطلبات والخطة الزمنية ويساهم بشكل فعال في توزيع المهام على أعضاء الفريق ومساعدتهم في حل المشاكل إن وجدت.



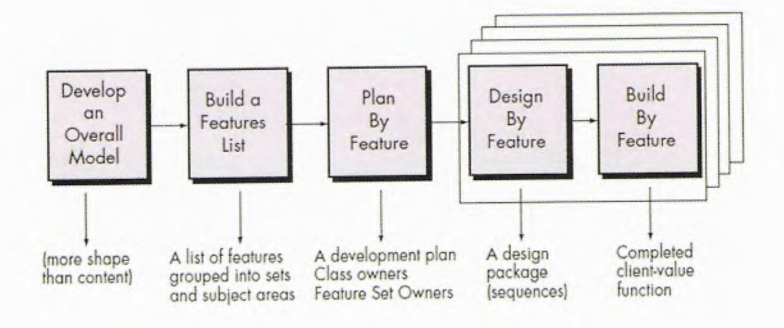
الشكل *(9)*: نموذج *Scrum*

3 - نماذج أخرى

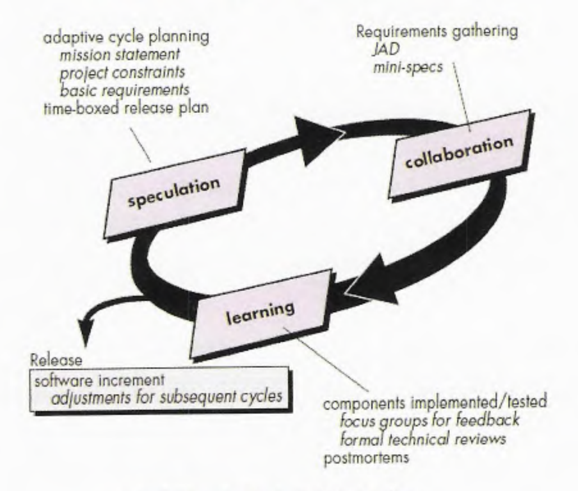
فيما يلي نستعرض أهم النماذج الرشيقة التي تتشابه فيما بينها في المبادئ العامة ولكن قد تختلف بالتفاصيل حيث يبين الشكل *(10)* نموذج التطوير المعتمد على الميزات *Feature driven development* الذي يعتمد على تقسيم المتطلبات إلى مجموعة من المميزات والتعبير عنها بشكل رسمي عن طريق نموذج محدد ومن ثم تطوير كل ميز على حدى بإصدار مستقل.

يبين الشكل *(11)* نموذج التطوير المتكيف *Adaptive development* والذي يعتمد بشكل أساسي على المحددات الضابطة لكل إصدار والتعبير عن المتطلبات وتقسيمها باستخدام صناديق زمنية *Time boxed* ومن ثم بناء التطبيق اعتماداً عن طريق تصميم المكونات البرمجية اللازمة *Components*.

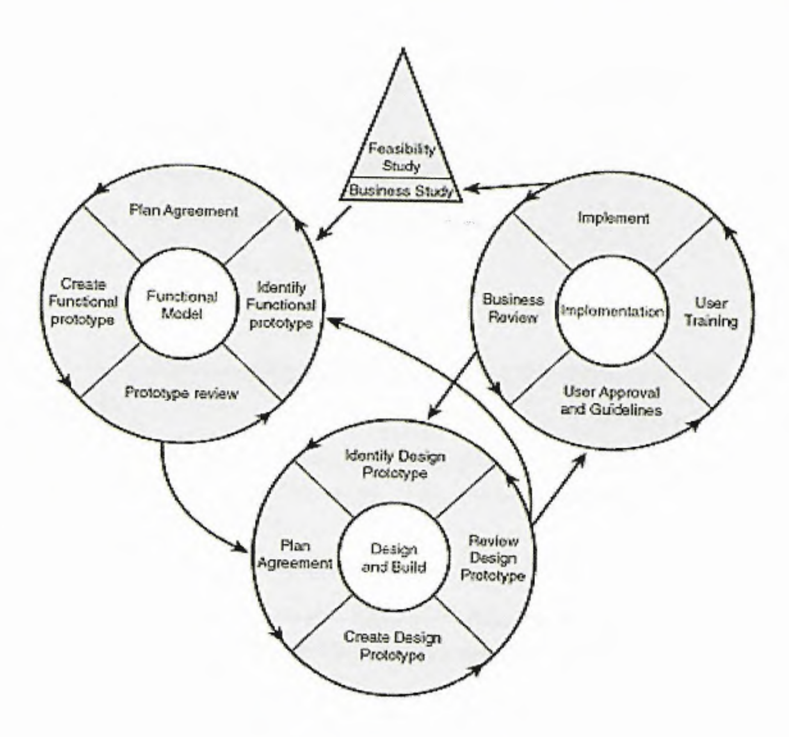
كما يبين الشكل *(12)* نموذج التطوير الديناميكي *Dynamic development* والذي يعتمد بشكل أساسي على التفاعل المستمر بين أطوار التطوير المختلفة وفي كافة الاتجاهات.



الشكل *(10)*: نموذج التطوير المعتمد على الميزات



الشكل *(11)*: نموذج التطوير المتكيف



الشكل (12): النموذج الديناميكي

2 - 1 - 6 - تمارين فصلية

1- يدعى النموذج المصغر غير الكامل للنظام البرمجي المراد تطويره:

a. النموذج الأولي *Prototype*.

b. نموذج علاقات الكائنات *Entity relationship model*.

c. نموذج تدفق المعطيات *Data flow model*.

d. جميع ما ذكر أعلاه.

2- يمكن فهم مصطلح هندسة البرمجيات *Software Engineering* على أنه:

a. مجموعة من النظريات والطرق والأدوات الهندسية المختصة في تطوير نظام برمجي جيد وسهل الاستخدام وضمن تكلفة معقولة.

b. مجموعة من البرمجيات المختصة في علم الهندسة.

c. مجموعة من الأدوات الهندسية المعروفة والمستخدمة في العديد من الاختصاصات ويمكن استثمارها برمجياً.

d. كل ما ذكر أعلاه.

3- تكون شكل عملية الانتقال بين أطوار النموذج الحلزوني لتطوير البرمجيات:

a. تناقصي.

b. إهليلجي.

c. تزايدي.

d. متعرج.

4- نماذج الإجرائيات *Process models* المستخدمة في تطوير الأنظمة البرمجية هي:

a. نماذج عامة يمكن استخدامها في تطوير أي نظام برمجي مهما اختلف نوعه.

b. نماذج خاصة تختلف باختلاف الأنظمة البرمجية المراد تطويرها.

c. نماذج ثابتة ومحددة لا يمكن تطويرها أو تعديلها من دون موافقة موجدها.

d. نماذج ينصح باستخدامها أثناء تطوير الأنظمة البرمجية ولكن يمكن الاستغناء عنها.

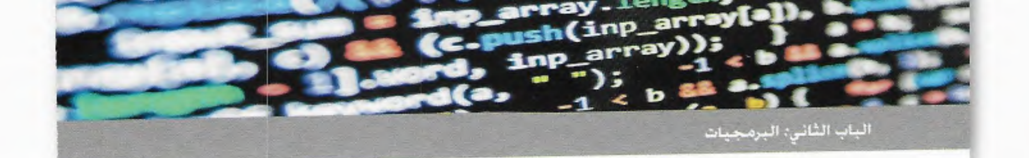
5- الرشاقة في تطوير البرمجيات *Agile Software Development*:

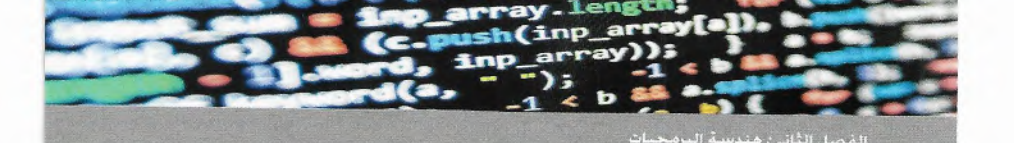
a. مجموعة من طرق تطوير البرمجيات على أساس التنمية المتكررة.

b. مجموعة من طرق تطوير البرمجيات التي تلبي الاحتياجات المتبدلة بشكل سريع وديناميكي.

c. مجموعة من طرق تطوير البرمجيات التي تهدف الى تسليم المنتج البرمجي بشكل سريع.

d. جميع ما ذكر أعلاه.





2 - 2 - هندسة المتطلبات *Requirement Engineering*

2 - 2 - 1 - المتطلبات *Requirements*

المتطلبات هي وصف لمجموعة خدمات النظام الوظيفية وغير الوظيفية والقيود التي يتم تحديدها خلال عملية التطوير البرمجي.

يمكن تمييز أنواع مختلفة للمتطلبات:

- **متطلبات المستخدم** *User requirement*: وتمثل البيانات المكتوبة باللغة الطبيعية بالإضافة إلى الرسوم البيانية لمجموعة الخدمات التي يوفرها النظام مع القيود التشغيلية بحيث تكون مكتوبة وموجهة للزبائن.
- **متطلبات النظام** *System requirements*: وتمثل وثيقة منظمة تحدد وصفاً تفصيلياً لوظائف النظام وخدماته والقيود التشغيلية بحيث تحدد ما يجب تنفيذه بحيث يكون جزءاً من عقد بين الزبون والمتعاقد.
- **المتطلبات الوظيفية** *Functional requirements*: ويعبر عن مجموعة الوظائف والخدمات التي ينبغي أن يوفرها النظام، وكيف ينبغي أن يتفاعل النظام مع مدخلات معينة وكيف ينبغي للنظام أن يتصرف في حالات معينة. قد يذكر في المتطلبات الوظيفية ما يجب على النظام عدم القيام به. تنشأ المشاكل عندما تكون المتطلبات الوظيفية غير دقيقة بحيث يتم تفسيرها بطرق مختلفة من قبل المطورين والمستخدمين لذا يجب أن تكون المتطلبات كاملة ومتسقة بحيث لا تحتوي تناقضات في أوصاف مرافق النظام.

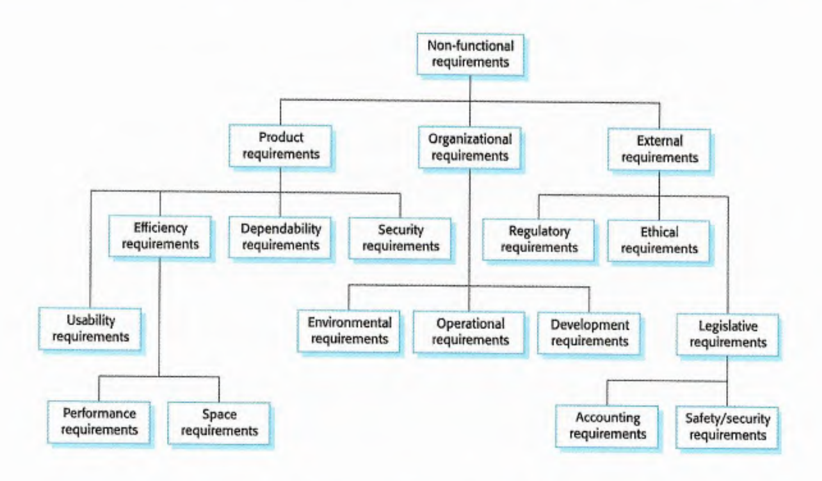
مثال: من المتطلبات الوظيفية لنظام مركز عيادات طبية:

- يجب أن يكون المستخدم قادرا على البحث في قوائم التعيينات لجميع العيادات.

- يجب على النظام توليد كل يوم، لكل عيادة، قائمة المرضى الذين من المتوقع أن يحضروا للمعاينة في ذلك اليوم.

- يتم تحديد كل موظف مستخدم للنظام بشكل فريد من قبل رقم الموظف المكون من *8* أرقام.

- المتطلبات غير الوظيفية *Non-functional requirements*: وهي تحدد خصائص النظام والقيود المفروضة على الخدمات أو الوظائف التي يقدمها النظام مثل قيود التوقيت، والقيود المفروضة على عملية التطوير، والمعايير، وما إلى ذلك. ويمكن أيضاً أن تحدد متطلبات العملية البرمجية المقادة ببيئة عمل محددة أو لغة البرمجة أو طريقة التطوير. غالباً ما يتم تحديد هذه المتطلبات على مستوى النظام ككل. وقد تكون المتطلبات غير الوظيفية أكثر أهمية من المتطلبات الوظيفية حيث يعتبر النظام غير مجدي إذا لم يتم الوفاء بها. يمكن تصنيف المتطلبات غير الوظيفية إلى عدة أصناف فرعية كما هو مبين في الشكل *(13)*:



الشكل *(13)*: المتطلبات غير الوظيفية

- **متطلبات المنتج** ***Product requirements:*** وهي مجموعة المتطلبات التي تحدد أن المنتج الذي تم تسليمه يجب أن يتصرف بطريقة معينة مثل تحديد سرعة التنفيذ والموثوقية.

- **المتطلبات التنظيمية** ***Organizational requirements:*** وهي مجموعة المتطلبات التي تنتج عن السياسات والإجراءات التنظيمية في الشركات.

- **المتطلبات الخارجية** ***External requirements:*** وهي مجموعة المتطلبات التي تنشأ عن عوامل خارجة عن النظام وعملية تطويره، ومتطلبات التشغيل البيني، والمتطلبات التشريعية، وما إلى ذلك.

مثال: من المتطلبات غير الوظيفية لنظام مركز عيادات طبية:

- **متطلبات المنتج:** يجب أن يكون النظام متاحاً لجميع العيادات خلال ساعات العمل العادية *(من الاثنين إلى الجمعة، 08.30-17.30)* يجب

ألا يتجاوز وقت التوقف عن العمل خلال ساعات العمل العادية خمس ثوان في أي يوم واحد.

- **المتطلبات التنظيمية:** يجب على مستخدمي النظام أن يوثقوا أنفسهم باستخدام بطاقة هويتهم الصحية.

- **المتطلبات الخارجية:** يجب على النظام تنفيذ أحكام خصوصية المريض حسب القوانين الناظمة.

2 - 2 - 2 - أصحاب المصلحة *Stakeholders*

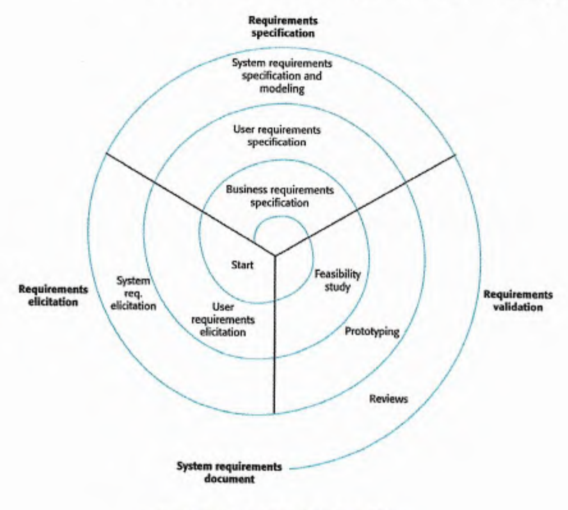
يدعى أي شخص أو مؤسسة يمكن أن تؤثر أو تستفيد من النظام المطور بطريقة ما بأصحاب المصلحة، ومن أشكالهم:

- **المستخدمون** *Users*: يقومون بتعريف وظائف النظام والاستفادة منه.
- **المطورين** *Developers*: مهمتهم بناء ونشر النظام بناءً على المواصفات.
- **مسؤولوا النظام** *System Administrators*: يقومون بتشغيل النظام بعد نشره.
- **المختبرون** *Testers*: يقومون باختبار النظام للتأكد من جاهزيته للاستخدام.
- **موظفي الدعم** *Support Staff*: تقديم الدعم لمستخدمي المنتج أو النظام عند تشغيله.
- **المشرفون** *Maintainers*: إدارة تطور النظام بمجرد تشغيله.
- **مديري النظام** *System managers*.
- **مالكي النظام** *System owners*.

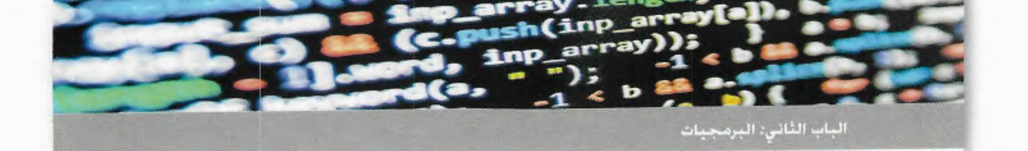
2 - 2 - 3 - هندسة المتطلبات *Requirement Engineering*

وهي مجموعة من الإجرائيات التي تتضمن جمع وتوثيق والتحقق من الخدمات التي يحتاجها الزبون من النظام المراد تطويره والقيود التي يعمل بموجبها. تختلف الإجرائيات المتبعة في هندسة المتطلبات عادة باختلاف التطبيق واختلاف الأشخاص المعنيين، ومع ذلك هناك عدد من الإجرائيات العامة المشتركة لجميع العمليات كما هو مبين بالشكل *(14)* وهي:

1. استيضاح المتطلبات *Requirement Elicitation*.
2. تحليل المتطلبات *Requirements Analysis*.
3. التحقق من صحة المتطلبات *Requirements validation*.
4. إدارة المتطلبات *Requirements management*.



الشكل *(14)*: هندسة المتطلبات



1 - استيضاح المتطلبات *Requirement Elicitation*

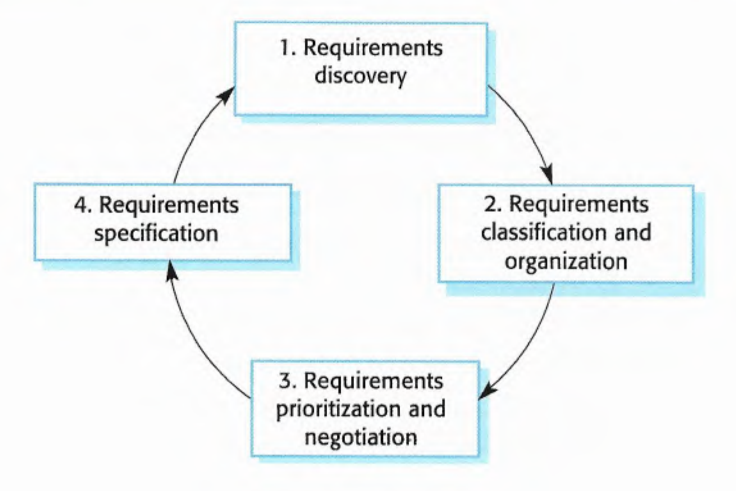
يعمل مهندسوا البرمجيات مع مجموعة من أصحاب المصلحة في النظام للتعرف على نطاق التطبيق، والخدمات التي ينبغي أن يوفرها النظام، وأداء النظام المطلوب، والقيود المفروضة على الأجهزة، والأنظمة الأخرى، وما إلى ذلك. تتضمن هذه المرحلة الإجرائيات التالية:

- **اكتشاف المتطلبات** ***Requirements discovery:*** وهي عبارة عن عمليات جمع المعلومات عن الأنظمة الموجودة والمطلوبة واستخلاص متطلبات المستخدم والنظام من هذه المعلومات، ويتم التفاعل مع أصحاب المصلحة في النظام بدءاً من المدراء وصولاً إلى المنظمين الخارجيين. ومن وسائل جمع المعلومات المستخدمة المقابلة *Interview*، الاستبيانات *Surveys*، المراقبة *Observation*.

- **تصنيف المتطلبات** ***Requirement Classifications:*** تجميع المتطلبات ذات الصلة في مجموعات وتنظيمها.

- **تحديد أولوية المتطلبات والتفاوض** ***Requirements prioritization and negotiation:*** ترتيب مجموعات المتطلبات حسب الأولويات المتاحة ويتم حل أي اختلاف بين أصحاب المصحلة عن طريق التفاوض.

- **مواصفات المتطلبات** ***Requirements specification:*** توثيق المتطلبات والمدخلات.



الشكل *(15)*: استنباط المتطلبات

2 - تحليل المتطلبات *Requirement Analysis*

يتضمن تحليل المتطلبات مايلي:

1 - شرح المتطلبات الأساسية المنصوص عليها في المهام الهندسية السابقة.

2 - بناء النماذج التي تصور سيناريوهات المستخدم والأنشطة الوظيفية وفئات المشكلة وعلاقاتها.

3 - يجب ان تتوفر في كتابة متطلبات المستخدم والنظام في وثيقة المتطلبات *(*المواصفات*)* الشروط التالية:

- يجب أن تكون متطلبات المستخدم مفهومة من قبل المستخدمين النهائيين والعملاء الذين ليس لديهم خلفية تقنية.

- متطلبات النظام هي متطلبات أكثر تفصيلاً وقد تشمل المزيد من المعلومات التقنية.

- قد تكون المتطلبات جزءاً من عقد لتطوير النظام.

- من المهم أن تكون كاملة قدر الإمكان.

4- تثبيت المتطلبات في وثيقة المواصفات *Specification* وهي البيان الرسمي لما هو مطلوب من مطوري النظام وينبغي أن تشمل كلاً من تعريف متطلبات المستخدم وتحديد مواصفات النظام. لا يمكن اعتبار وثيقة المواصفات وثيقة تصميمية وينبغي قدر الإمكان أن يحدد ما يجب أن يفعله النظام عوضاً عن كيفية القيام بذلك. يمكن أن تستخدم هذه الوثيقة اللغة الطبيعية، لغة تعبيرية منظمة بقالب *Template*، النماذج الرسومية *Models*، أو اللغات الرسمية الرياضية *Formal languages* في عمليات التوثيق.

3 - التحقق من صحة المتطلبات *Requirement Validation*

ومجموعة من الإجرائيات المعنية بإثبات أن متطلبات النظام هي ما يريده الزبون بالفعل وتعتبر عملية التحقق من صحة المتطلبات مهمة جداً بسبب أن تكاليف خطأ المتطلبات عالية حيث يمكن أن يكلف إصلاح خطأ المتطلبات بعد التسليم ما يصل إلى *100* مرة من تكلفة إصلاح خطأ في التنفيذ.

من أهم الخصائص التي يجب أن تتوفر في المتطلبات: الصلاحية *validity* حيث يتم التحقق من إمكانية تلبية النظام لاحتياجات الزبون، التناسق *consistency* حيث يتم مراجعة المتطلبات لاكتشاف أي اختلافات أو تعارضات فيمابينها، الاكتمال*completnace* حيث يتم معرفة شمولية المتطلبات لجميع الوظائف ، الواقعية *reality* حيث يتم دراسة إمكانية تنفيذ المتطلبات بالنظر إلى الميزانية والتكنولوجيا المتاحة.

من أهم الطرق المستخدمة للتحقق من صحة المتطلبات طرق المراجعة الرسمية *Formal reviews* حيث يتم إجراء مراجعة منتظمة أثناء وبعد صياغة تعريف المتطلبات حيث يشارك كل من الزبائن والموظفين المتعاقدين في عمليات التحقق.

4 - قرارات إدارة المتطلبات *Requirements management decisions*

تتضمن إجرائيات إدارة المتطلبات مايلي:

- **تحديد المتطلبات:** يجب تحديد كل متطلب على حدى بشكل مميز بحيث يمكن الرجوع إليها مع المتطلبات الأخرى.
- **عملية إدارة التغيير:** هي مجموعة الأنشطة التي تقيم تأثير التغييرات في المتطلبات وتكلفتها حيث يتم تقييم تأثير التغييرات على المشروع بأكمله.
- **سياسات التتبع:** تحدد هذه السياسات العلاقات بين كل متطلب وبين المتطلبات الأخرى وتحاول أن تربط بين كل متطلب والمكون التصميمي والبرمجي الذي يلبيه.
- **دعم الأدوات:** الأدوات التي يمكن استخدامها تتراوح بين أنظمة إدارة المتطلبات المتخصصة وجداول البيانات وأنظمة قواعد البيانات البسيطة.

وتعتبر إ دارة المتطلبات عملية تتم من خلالها إدارة المتطلبات المتغيرة خلال عملية هندسة المتطلبات وتطوير النظام. وتظهر متطلبات جديدة مع تطور النظام وبعد دخوله حيز الاستخدام.

2 - 2 - 4 - تمارين فصلية

1- من الأشخاص الأقل وجوداً عند تطوير أي مشروع برمجي:

a. مدير المشروع *Project manager*.

b. المحلل *Analyst*.

c. الزبون *Customer*.

d. المستخدم النهائي *End user*.

2- جمع متطلبات النظم البرمجية تكون عادة أحد مهام:

a. المبرمج *Programmer*.

b. الزبون *Customer*.

c. المحلل *Analyst*.

d. المصمم *Designer*.

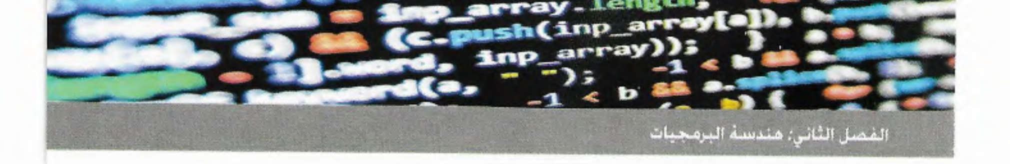
3- أي مايلي ليس من أهداف بناء نموذج متطلبات النظام البرمجي المراد تطويره *Requirement Model*؟

a. تحديد متطلبات النظام البرمجي بطريقة سهلة الفهم والاختبار.

b. وصف متطلبات الزبائن بصورة دقيقة.

c. تطوير حلول جيدة لجميع المشكلات البرمجية التي تم تحديدها.

d. وضع أسس واقعية وصحيحة لبداية تصميم النظام.



4- أي مايلي ليس من أطوار هندسة المتطلبات:

a. استيضاح المتطلبات.

b. إدارة المتطلبات.

c. توثيق المتطلبات.

d. تنفيذ المتطلبات.

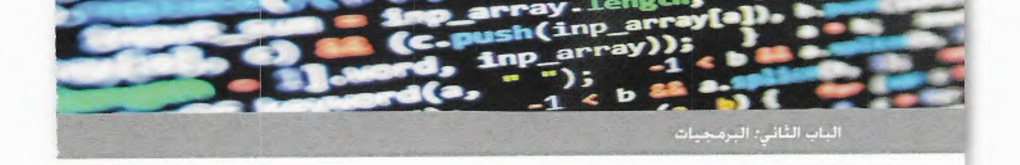
5- أشر إلى العبارة الصحيحة مايلي:

a. تغيير المتطلبات غير متاحة بعد الاتفاق وتوقيع العقد.

b. تغيير المتطلبات متاحة فقط قبل الاتفاق وتوقيع العقد.

c. تغيير المتطلبات متاحة قبل وبعد الاتفاق وتوقيع العقد.

d. تغيير المتطلبات غير متاحة بالمطلق.



2 - 3 - نمذجة النظام *System Modelling*

2 - 3 - 1 - أساسيات نمذجة النظام *System Modelling*

نمذجة النظام هي عملية تطوير نماذج مصغرة عن النظام باستخدام نوع من الرسومات البيانية عن طريق لغة نمذجة معينة مثل لغة النمذجة الموحدة *(Unified Modelling Language) (UML)* للمساعدة على على فهم وظائف النظام وتأمين صلة تواصل بين الزبون والمطورين من جهة وبين المطورين فيما بينهم من جهة أخرى.

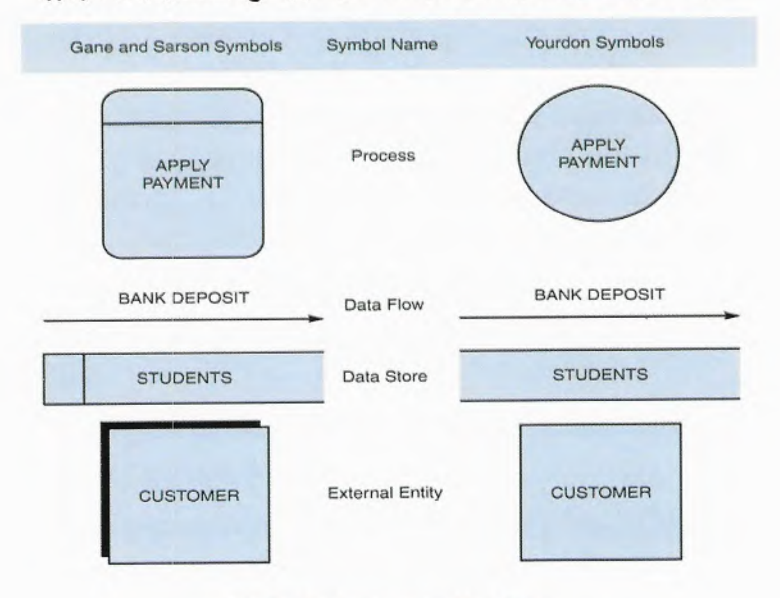
يمكن بناء نماذج النظام على عدة مستويات لإعطاء صورة متكاملة عنه:

- **نماذج السياق** ***Context models***: توضح حالات التفاعل المختلفة بين النظام والبيئة المحيطة.
- **نماذج التفاعل** ***Interaction models***: توضح حالات التفاعل المختلفة بين مكونات النظام فيما بينها وكذلك مع البيئة المحيطة.
- **النماذج الهيكلية** ***Structural models***: توضح المكونات التركيبية والبنيوية التي يتألف منها النظام.
- **النماذج السلوكية** ***Behavioral diagrams***: توضح السلوك الديناميكي للنظام وكيف استجابته للأحداث.

2 - 3 - 2 - نماذج السياق *Context Models*

تستخدم نماذج السياق لتوضيح السياق التشغيلي لنظام ما وهي تبين ما يقع خارج حدود النظام سواء من أنظمة أخرى تستخدم أو تعتمد على النظام الجاري تطويره أو مستخدمين عاديين. من المخططات

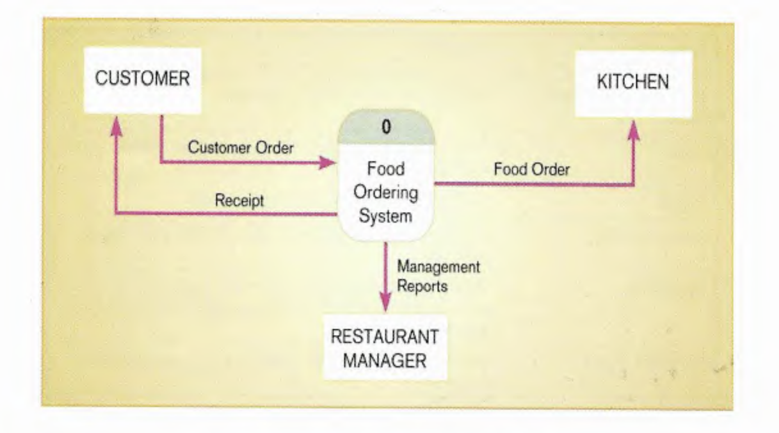
التي تستخدم لتوضيح المنظور السياقي للنظام هي مخططات تدفق البيانات *Data flow diagram (DFD)* حيث يوضح الشكل *(16)* الرموز



الشكل *(16)*: رموز مخطط تدفق البيانات

مثال عن استخدام *DFD* للتعبير عن المنظور السياقي للنظام المراد بناؤه

المراد بناء نظام طلبات الطعام في مطعم ما بحيث يتلقى طلبات الزبائن ليرسلها إلى المطبخ من أجل إعداد الوجبة كما يولد التقارير المناسبة للإدارة. يوضح الشكل *(17)* المخطط السياقي لهذه النظام.



الشكل *(17)*: النموذج السياقي لنظام المطعم

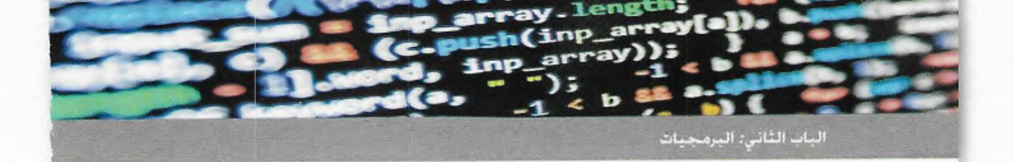
2 - 3 - 3 - نماذج التفاعل ***Interaction Models***

تعتبر نماذج التفاعل من النماذج المساعدة في تحديد وتوضيح متطلبات المستخدم. من المخططات المستخدمة لهذا الغرض مخططات حالة الاستخدام *Use case diagrams* التي تعتبر من أحد نماذج *UML*. يتألف نموذج حالة الاستخدام من الرموز التالية:

1. **حالة الاستخدام *Use case*:** مجموعة فرعية من وظائف النظام العام يتم تمثيلها بقطع ناقص أفقي مع اسم حالة الاستخدام داخل القطع الناقص كجملة فعلية.

Use Case
Symbol

2. **سيناريو حالة الاستخدام:** وصف نصي لهدف الأعمال وكيفية تفاعل المستخدم مع النظام لإنجاز المهمة كماهو مبين في الشكل *(18)*.



Property	Definition
Business Use Case Name	Defines the name of the use case.
Actor	Recipient of the service. Must lie outside the business boundary.
Trigger	Initiating event of the business process.
Pre-conditions	Conditions that must be satisfied for the use case to take place.
Basic Flow	Description of the flow of activities that ordinarily take place for the execution of the process defined in the use case.
Alternate Flows	Description of alternate courses of execution of the process.
Post-conditions	Conditions that must hold true after the termination of the process.

الشكل *(18)*: سيناريو حالة الاستخدام



Actor Symbol

3. **الجهة الفاعلة*Actors*:** أي شخص أو أي جهة تتفاعل مع النظام لتبادل المعلومات كالإنسان، منظمة، نظام معلومات آخر، جهاز خارجي، حتى الوقت. يمكن أن يكون فاعل النظام من أحد الأنواع التالية:

- **الفاعل التجاري الرئيسي *Primary Business Actor***: وهو من أصحاب المصلحة التي تستفيد في المقام الأول من تنفيذ حالة الاستخدام ومثالها الموظف الذي يحصل على الراتب.
- **فاعل النظام الأساسي *Primary System Actor***: وهو من أصحاب المصلحة التي تتفاعل مباشرة مع النظام لبدء أو إطلاق الأعمال، ومثالها إدخال معلومات الإيداع من قبل أمين صندوق البنك.

- **فاعل خادم خارجي** ***External Server Actor:*** وهو من أصحاب المصلحة الذين يستجيبون لطلب من حالة الاستخدام ومثالها مكتب الائتمان الذي يتحقق من رسوم بطاقة الائتمان.
- **فاعل استقبال خارجي** ***External Receiver Actor:*** وهو من أصحاب المصلحة التي ليست الجهة الفاعلة الرئيسية ولكنها تتلقى شيئاً ذا قيمة من حالة الاستخدام ومثالها أمين المستودع الذي يتلقى من النظام أمر تعبئة منتج ما.

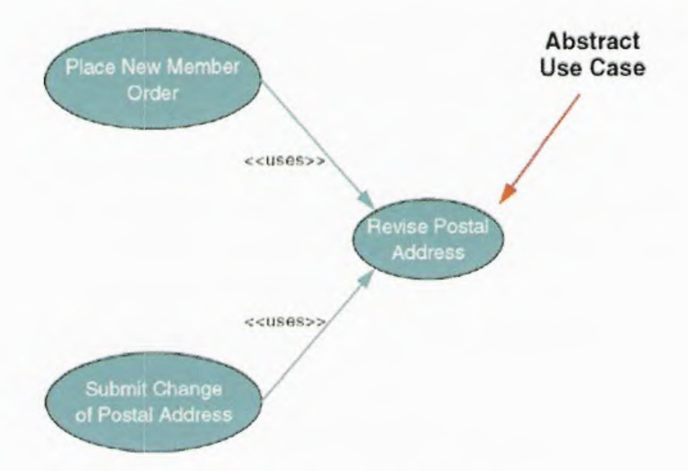
4. **العلاقات** ***Relations:*** تستخدم لربط رموز المخطط ببعضها ويمكن تصنيفها كمايلي:

- **التجميع** ***Association:*** وهي العلاقة التي تجمع بين الجهة الفاعلة وحالة الاستخدام كما هو موضح في الشكل *(19)*.



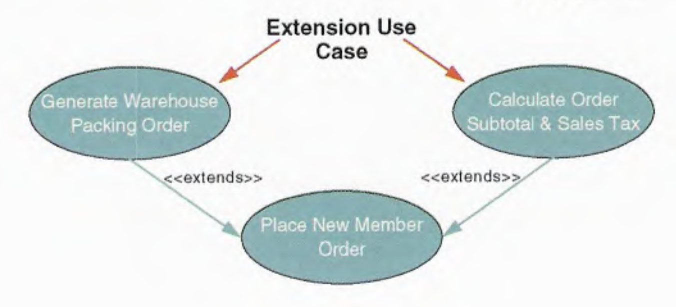
الشكل *(19)*: التجميع *Association*

- **التجريد** ***Abstraction:*** تستخدم للربط بين حالة الاستخدام وحالة استخدام أخرى مشتقة من أكثر من حالة استخدام بحيث تجمع بين الخطوات الشائعة الموجودة في كل منهم وهي علاقة إجبارية الحدوث كما هو مبين في الشكل *(20)*.



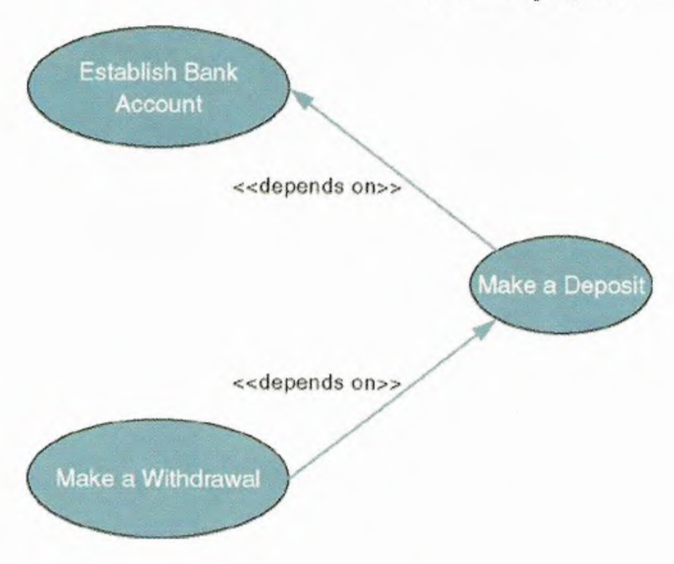
الشكل *(20)*: التجريد *Abstraction*

- **الامتداد** ***Extension:*** تستخدم للربط بين حالة الاستخدام وحالة استخدام أخرى مشتقة منها بحيث تكون امتداد للخطوات الموجودة في الأولى وهي علاقة اختيارية الحدوث كما هو مبين في الشكل *(21)*.



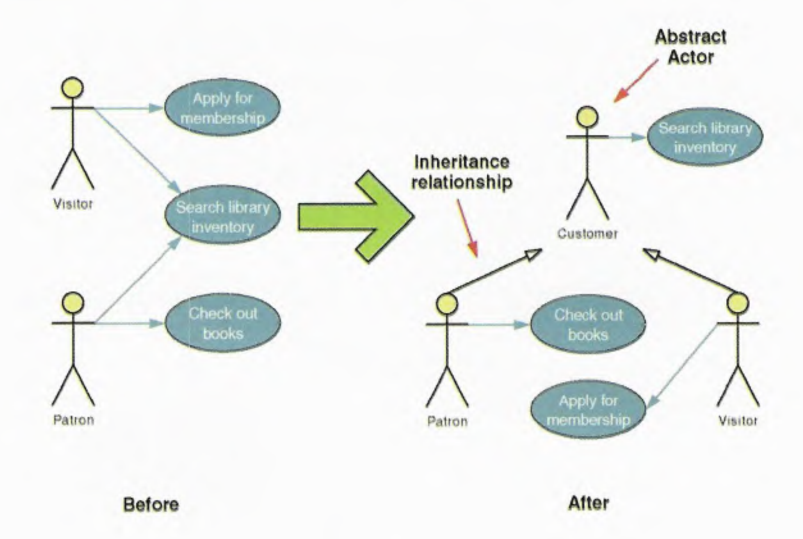
الشكل *(21)*: الامتداد *Extension*

- **الاعتمادية *Depends On*:** توضح درجة اعتمادية تنفيذ أحد حالات الاستخدام على غيرها حيث يمكن أن يساعد في تحديد التسلسل الذي تحتاج إليه حالات الاستخدام في عملية التنفيذ كما هو موضح في الشكل *(22)*.



الشكل *(22)*: الاعتمادية *Depends On*

- **الوراثة *Inheritance*:** تعبر عن علاقة حالة الاستخدام والحالات التي ترثها أو بين فاعل والفاعلون الذين يرثون منه كما هو موضح في الشكل *(23)*.



الشكل *(23)*: الوراثة *Inheritance*

يمكن تلخيص الخطوات اللازمة لبناء مخطط حالة الاستخدام بمايلي:

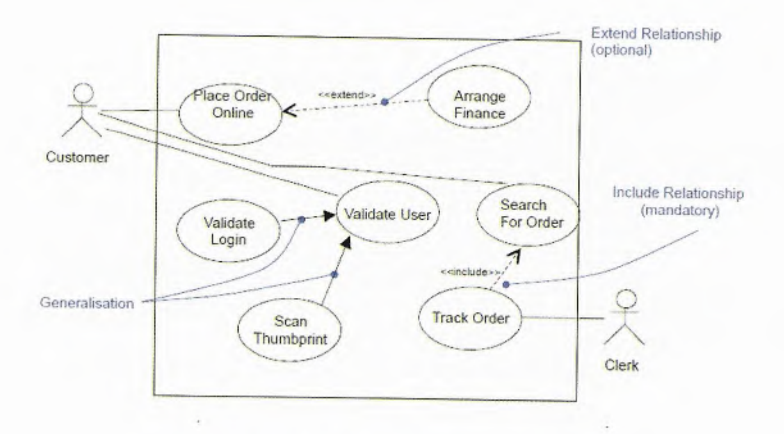
1. استخلاص وتحليل معلومات كافية عن المتطلبات.

2. تحديد الجهات الفاعلة.

3. تحديد حالات الاستخدام لكل جهة فاعلة.

4. بناء سيناريو الاستخدام لكل حالة استخدام.

وكمثال عن ذلك يبين الشكل *(24)* مثال عن مخطط حالة الاستخدام لنظام بيع الكتروني:



الشكل (24): مخطط الحالة جزئي عن نظام بيع الكتروني

2 - 3 - 4 - النماذج السلوكية *Behavioral Models*

توضح السلوك الديناميكي للنظام كما هو قيد التنفيذ وكيف استجابته للأحداث والبيانات وتظهر هذه النماذج تسلسل الإجراءات والتفاعلات بين مكونات النظام لتوليد ناتج مرتبط بها. من المخططات المستخدمة لهذا الغرض مخططات النشاط *Activity diagrams* والمخططات التتابع *Sequence diagrams* ومخططات التعاون *Collaboration diagrams* ومخططات الحالة *State diagrams* وكلها تعتبر من نماذج *UML*.

مخطط النشاط *Activity Diagram*

يمكن استخدام مخطط النشاط لرسم بياني لتدفق الأعمال وخطوات حالة الاستخدام باستخدام رموز

Business Process Modelling Notations (BPMN)

والتي يمكن تصنيفها إلى عدة مجموعات:

1 - عناصر التدفق *Flow objects*

Events

- **الحدث *Event*:** شيء يحدث أثناء معالجة سير العمل، ويقتصر في *BPMN* فقط على الأحداث التي تؤثر على تسلسل أو توقيت العملية.

Activities

- **النشاط *Activity*:** عمل أو نشاط يتم حدوثه أثناء تدفق الأعمال

Gateways

- **البوابة *Gateway*:** العناصر التي تتحكم في كيفية تفاعل تسلسل التدفقات لتحديد إمكانية دمجها أو تفريعها ومنها:



- ***Exclusive Decision / Merge*:** يشير إلى المواقع حيث يمكن أن يأخذ تدفق التسلسل مسارين أو أكثر من المسارات البديلة، ويمكن اتخاذ واحد فقط من المسارات.



- ***Parallel Fork / Join*:** توفير آلية لمزامنة تدفق مواز وخلق تدفق مواز.

2 - عناصر التوصيل *Connecting objects*

Sequence Flow

Message Flow

Association

تستخدم أساساً توصيل عناصر التدفق ومن أنواعها:

- تسلسل التدفق *Sequence Flow*.
- تدفق الرسالة *Message Flow*.
- الربط المتجانس *Association*.

Swimlanes - 3

تستخدم لتقسيم عناصر الرسم التخطيطي ومن أنواعها:

- **التجمع** ***Pool***: يمثل أحد المشاركين في العملية، على سبيل المثال: الشركة، الشخص، الدور، الخ.
- **الممر** ***Lane***: يمثل قسم فرعي داخل تجمع يستخدم لتنظيم الأنشطة وفئاتها.

Pool

Name

Lanes (within a Pool)

Name Name Name

Artefacts - 4

تستخدم لتوفير معلومات إضافية ولا ترتبط مباشرة بتدفق العملية ومن أنواعها:

- **عنصر البيانات** ***Data object***: يمثل البيانات المطلوبة من الأنشطة.
- **التعليقات التوضيحية** ***Annotation***: تمثل آلية لتمكين مصمم النماذج من تقديم معلومات إضافية كنص.
- **المجموعة** ***Group***: آلية لتجميع العناصر بشكل غير رسمي.

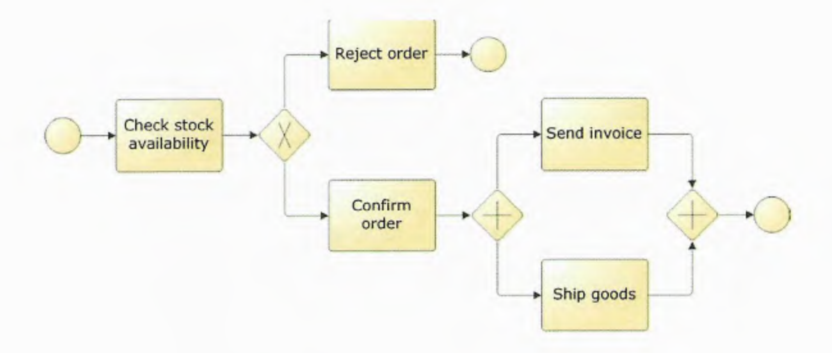
Data Object

Name [State]

Text Annotation

Add Text Here

Group



الشكل *(25)*: مثال عن مخطط النشاط

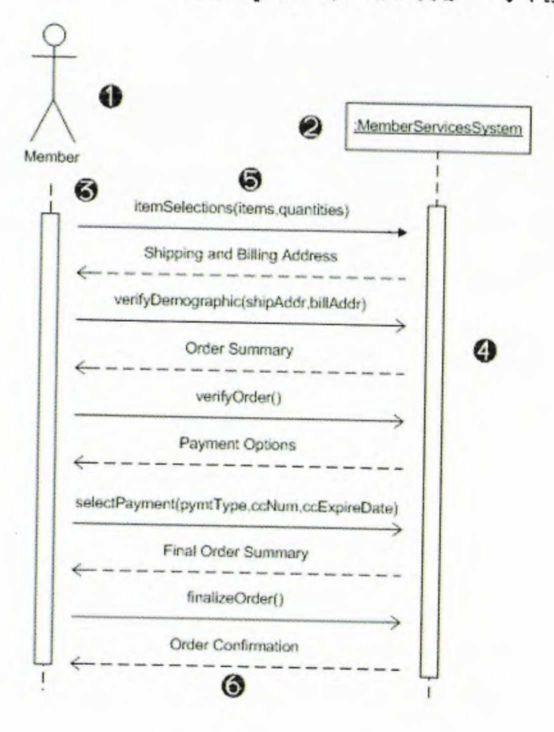
مخطط التتابع *Sequence Diagram*:

رسم بياني يصور التفاعل بين الفاعل ومكونات النظام لسيناريو حالة الاستخدام كما يساعد على تحديد الرسائل عالية المستوى التي تدخل في النظام وتخرج منه. تصنف رموز مخططات التتابع حسب ماهو مبين في الشكل *(26)* كمايلي:

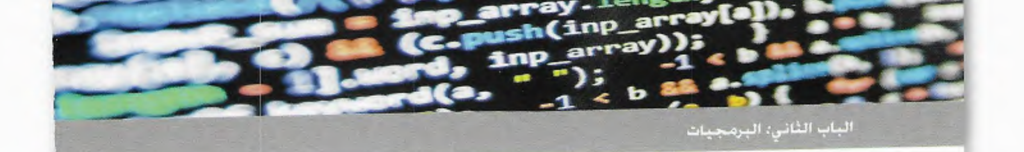
1. **فاعل النظام** ***System Actor*** .
2. **كائنات النظام** ***Objects***.
3. **خط الحياة** ***Lifeline***: ويمثل الفترة من خلق الكائن في الذاكرة إلى حال هدمه.
4. **شريط التفعيل** ***Activation bar***: ويمثل الفترة التي يكون فيها الكائن في حالة نشاط عند إرسال أو استقبال الرسائل.

5. **رسائل الدخل*input messages*:** تشير الأسهم الأفقية من الفاعل إلى النظام إلى رسائل الدخل.

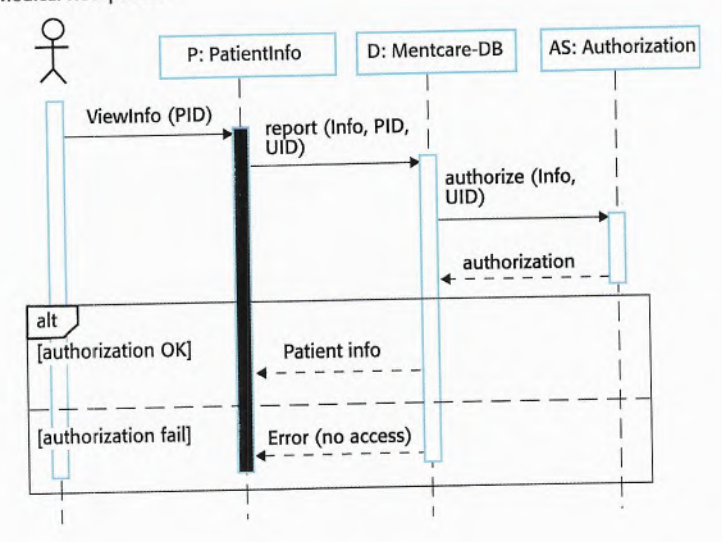
6. **رسائل الخرج** *output messages*: تشير الأسهم الأفقية من النظام إلى الفاعل بخطوط متقطعة إلى رسائل الخرج ويمكن أن تمثل نماذج الويب والتقارير والبريد الإلكتروني وغيرها.



الشكل *(26)*: رموز مخطط التسلسل



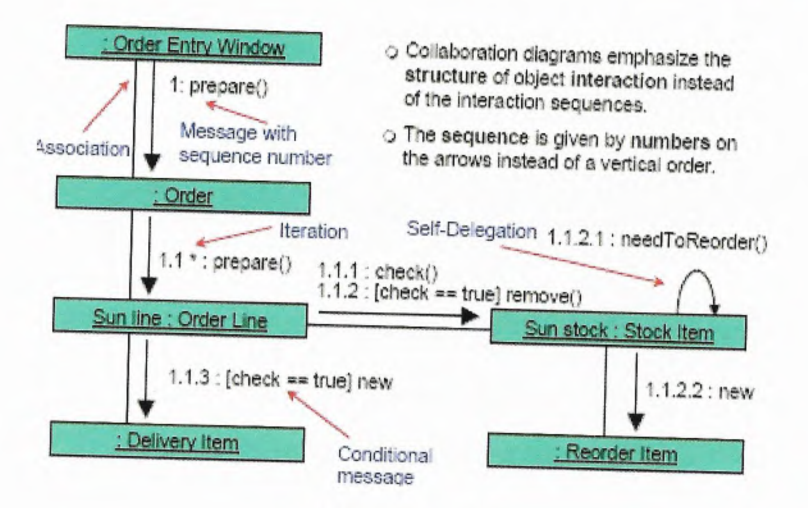
يبين الشكل *(27)* مثال عن مخطط التتابع لحالة البحث عن



الشكل *(27)*: مخطط التتابع

مخطط التعاون *Collaboration Diagram*

يمثل هذا المخطط السلوك الديناميكي للكائنات ويمكن أن يؤدي نفس الدور الذي يقوم به مخطط التفاعل إلا أن مخطط التعاون يركز أكثر على هيكلية الكائنات وترابطها أكثر من تسلسل الرسائل والتفاعلات فيما بينها. يبين الشكل *(28)* مثال عن مخطط التعاون لعملية الشراء أونلاين.

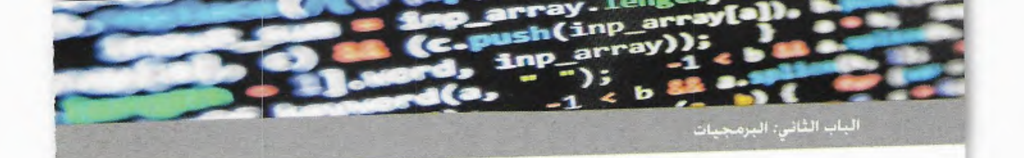


الشكل *(28)*: مخطط التعاون

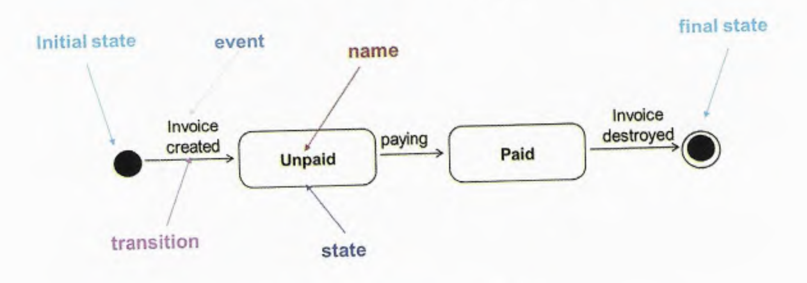
مخططات الحالة *State Diagrams*

يصف مخطط الحالة تسلسل العمليات على كائن محدد والتي تحدث استجابة للمؤثرات الخارجية من أحداث وبيانات وغيرها. يمكن تصنيف رموز مخططات الحالة كمايلي:

1. **الأحداث** *Events*: الحدث هو حدوث شيء ما في لحظة ما ناتج عن فعل ما كأن يقوم المستخدم بالضغط على الزر الأيسر للفأرة.
2. **الحالات** *States*: الحالة هي التي يجد فيها الكائن نفسه في أي لحظة.
3. **الانتقالات** *Transitions*: تأخذ الكائن من حالة إلى أخرى.
4. **الإجراءات** *Actions*: تحدث نتيجة لعملية الانتقال.



يمثل الشكل *(29)* مثال عن مخطط الحالة لفاتورة.

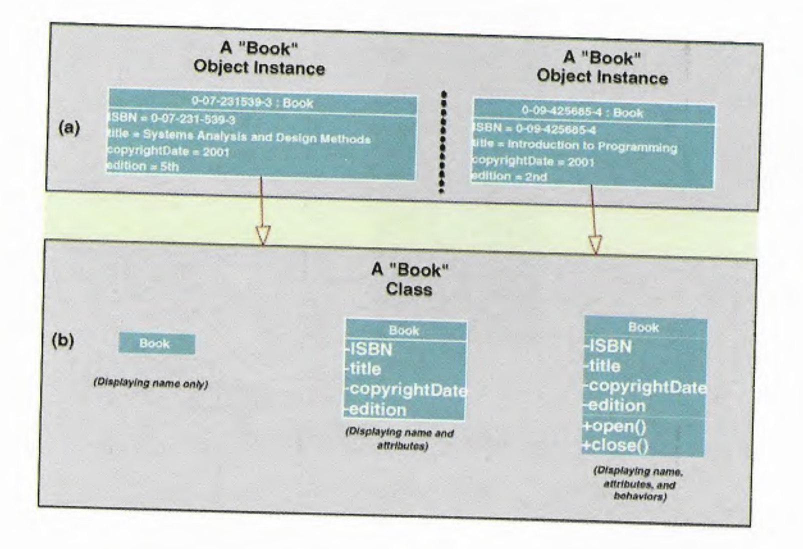


الشكل *(29)*: مخطط الحالة لفاتورة

2 - 3 - 5 - النماذج الهيكلية *Structure Modelling*

تقوم النماذج الهيكلية للبرمجيات مكونات النظام والعلاقة فيما بينهم. من المخططات المستخدمة لهذا الغرض مخطط الصفوف *Class diagram* والذي يعتبر من مخططات *UML*.

تستخدم مخططات الصفوف عند تعريف مكونات النظام حيث يعتبر كل صف كما هو موضح في الشكل *(30)* بمثابة توصيف عام لتجميع *Encapsulating* مجموعة من كائنات متشابهة في الصفات والسلوك *Objects* ويمكن أن تعني شيئاً في العالم الحقيقي، مثل المريض، وصفة طبية، طبيب وغير ذلك. يمكن تمييز كل كائن عن طريق مجموعة من السمات والصفات *Attributes* وكذلك عن طريق سلوكه *Behavior* الذي يعرف مجموعة من العمليات التي قد تؤثر على تعديل صفات هذا الكائن.

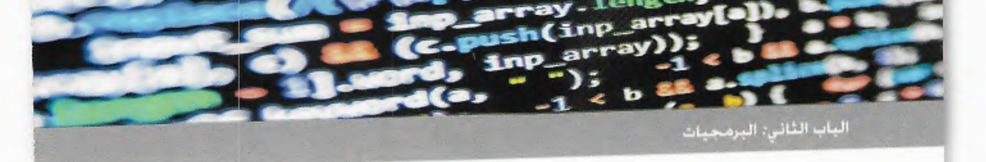


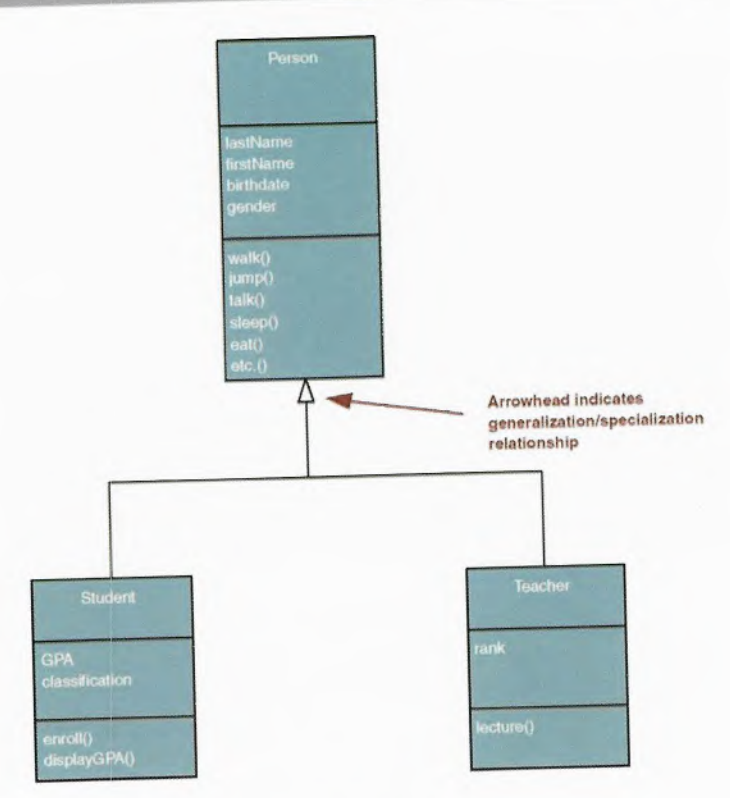
الشكل *(30)*: الصفوف

يتميز مخطط الصفوف بالأنواع المختلفة من العلاقات التي تربط الصفوف ببعضها وهي كالتالي:

1 - علاقة الوراثة *Inheritance* ا علاقة التعميم *Generlization*

وهي علاقة تربط الصفوف ذات صفات وسلوكيات موروثة تدعي *subtypes* بصف آخر يدعي بالأب أو السوبر *supertype* كما هو موضح في الشكل *(31)*.

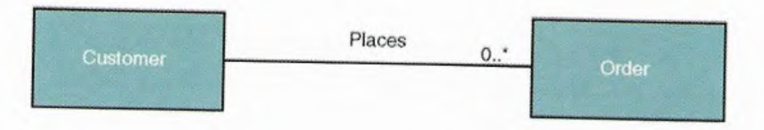




الشكل *(31)*: علاقة التعميم

2 - علاقة الارتباط *Association relationship*

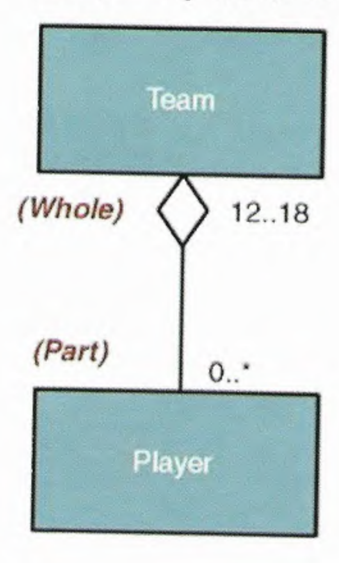
وهي العلاقة التي تربط بين كائنات النظام. وهي تعبر عن الحد الأدنى والحد الأقصى لعدد الكائنات من صف ما مع كائنات أخرى ذات صلة نتيجة استجابة لحدث ما كما هو مبين في الشكل *(32)*.



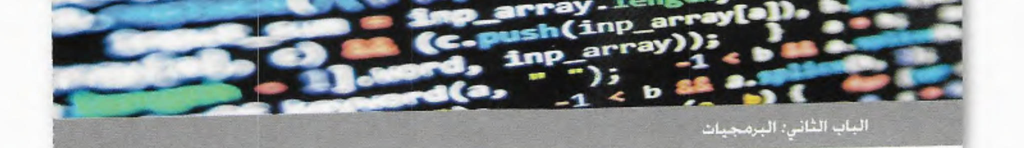
الشكل *(32): Multiplicity*

3 - التجميع *Aggregation*

وهي العلاقة بين عنصر واحد كبير و "كامل" يحتوي على واحد أو أكثر من العناصر الصغيرة، بشرط ألا تكون حياة هذه العناصر متعلقة بالعنصر الكامل كما هو مبين في الشكل *(33)*.

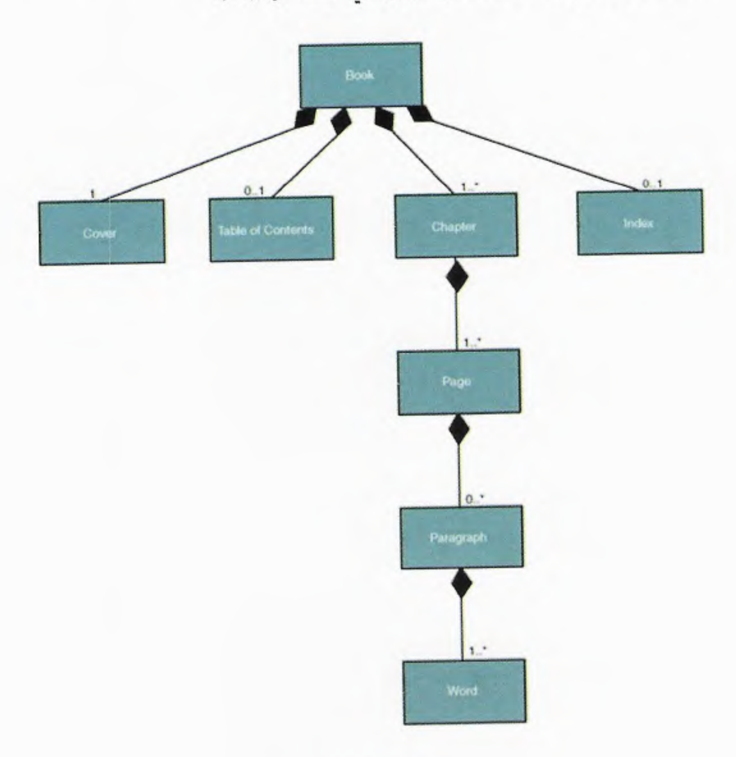


الشكل *(33)*: التجميع



4 - التركيب *Composition*

وهي العلاقة بين عنصر واحد كبير و "كامل" يحتوي على واحد أو أكثر من العناصر الصغيرة. وتكون حياة هذه العناصر متعلقة بالعنصر الكامل حيث إن تدمر العنصر الكامل تدمر كل العناصر الصغير المكونة له كما هو مبين في الشكل *(34)*.



الشكل *(34)*: التركيب

2 - 3 - 6 - تمارين فصلية

1- المصطلح *UML* هو اختصار لــ:

a. منطق النمذجة الغير المحدد *Undefined Modelling Logic*.

b. لغة النمذجة الموحدة *Unified Modelling Language*.

c. لغة النمذجة الغير محددة *Undefined Modelling Language*.

d. منطق النمذجة الموحد *Unified Modelling Logic*.

2- يمكن أن تشاهد علاقة الوراثة *Inheritance* في إحدى المخططات التالية:

a. مخطط الصفوف *Class diagram*.

b. مخطط تدفق المعطيات *Data flow diagram*.

c. مخطط الأنشطة *Activity diagram*.

d. مخطط التسلسل *Sequence diagram*.

3 - يستخدم مخطط الأنشطة *Activity Diagram* في طور تحليل النظام البرمجي لوصف:

a. سلوك النظام *Behaviour*.

b. كائنات النظام *Objects*.

c. تفاصيل تخاطب عناصر النظام *Scenario*.

d. حركة المعطيات داخل النظام *Flow*.

4 - أي مما يلي غير موجود في جدول سرد حالة الاستخدام *Use case Narrative*؟

a. اسم حالة الاستخدام *Use case Name*.

b. رقم حالة الاستخدام *Use case Number*.

c. فاعل حالة الاستخدام الرئيسي *Primary Actor*.

d. خطوات تنفيذ حالة الاستخدام *Main Flow*.

5 - يطلق على القيد الواجب فكه من قبل حالة استخدام محددة *Use case* قبل أن يتم تفعيلها:

a. الشرط الأولي *Precondition*.

b. القادح *Trigger*.

c. الشرط النهائي *Post condition*.

d. النتيجة *Conclusion*.

6 - يتم نمذجة الرسائل المتبادلة بين كائنات النظام البرمجي *objects* باستخدام:

a. مخطط حالات الاستخدام *Use case diagram*.

b. مخطط الأنشطة *Activity diagram*.

c. مخطط التسلسل *Sequence diagram*.

d. مخطط تدفق المعطيات *Data flow diagram*.

2 - 4 - تصميم النظام *System Design*

2 - 4 - 1 - تصميم النظام *System Design*

وهو عبارة عن وصف وتنظيم وبناء جميع مكونات النظام، يحدد فيه المعمارية ومستوى مفصل للعناصر. الغرض الأساسي هو تمكين بناء النظام ونشره. يمتلك تصميم النظام مستويين اثنين من مهام الاختصاص:

1- التصميم المعماري (مستوى عالي) *Architectural Design*:

يتضمن تحديد الأجهزة والبرمجيات والبنية التحتية للنظام ويتعلق بفهم كيفية تنظيم نظام البرمجيات وتصميم الهيكل العام لذلك النظام وهو الرابط الحاسم بين التصميم وهندسة المتطلبات، لأنه يحدد المكونات الهيكلية الرئيسية في النظام والعلاقة بينهما.

2- التصميم التفصيلي (مستوى منخفض) *Detail Design*:

يركز على تحديد الوحدات الصغيرة مثل تصميم البرمجيات لحالة استخدام معينة ويوضح التجريدات الرئيسية في النظام كعناصر أو فئات العنصر أو أجهزة وكيفية توزيعها كما يبين - في وقت التشغيل- العمليات المتفاعلة داخل النظام.

2 - 4 - 2 - أنشطة التصميم *Design Activities*

1- تصميم خدمات الدعم *Design Support Services*:

- دمج الأنظمة الجديدة في النظم القائمة.
- تثبيت خدمات الدعم لأول مرة.
- استبدال الأنظمة الحالية.

2- تصميم هندسة البرمجيات *Design the Software Architecture*:

- تقسيم البرامج إلى مكونات.
- توزيع المكونات عبر منصات المعالجة.
- تحديد الأنماط المعمارية المناسبة.
- توزيع مخططات الصفوف على طبقات البرمجيات من حيث:
  - تحديد مكان تنفيذ الطبقات وطرق الاتصال.
  - تحديد لغة (لغات) البرمجة اللازمة.

3- تصميم إنجاز حالة الاستخدام *Design Use Case Realizations*:

- تحديد جميع تفاعلات الكائنات التي تدعم حالة استخدام معينة.
- تحديد التفاعلات بين البرامج والمستخدمين والجهات الفاعلة في الأنظمة الخارجية.

4- تصميم قاعدة البيانات *Design the Database*.

5- تصميم واجهات النظام والمستخدم *Design the System and User Interfaces*.

2 - 4 - 3 - الأنماط المعمارية *Architectural Patterns*

- الأنماط هي وسيلة لتمثيل المعرفة ومشاركتها وإعادة استخدامها.
- النمط المعماري هو وصف معين لممارسة التصميم الجيد والتي تم تجربتها واختبارها في بيئات مختلفة.
- يجب أن تتضمن الأنماط معلومات حول متى تكون ومتى لا تكون مفيدة.
- يمكن تمثيل الأنماط باستخدام الجداول والرسوم البيانية.

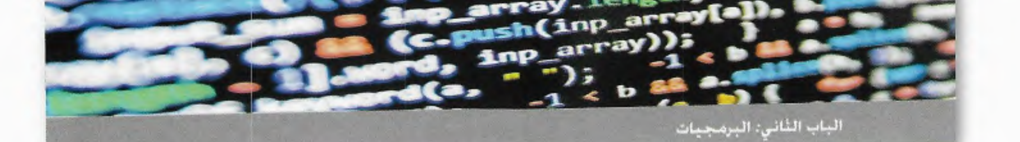
من أهم الأنماط المعمارية المستخدمة في تصميم التطبيقات البرمجية:

1 - نمط الطبقات Layered pattern

يبين الشكل *(35)* بنية الطبقات حيث تقسم فيه بنية النظام إلى مجموعة من الطبقات كل منها يوفر مجموعة من الخدمات بحيث تدعم التطوير التدريجي للنظام في طبقات مختلفة وعندما تتغير واجهة طبقة تتأثر الطبقة المجاورة فقط.

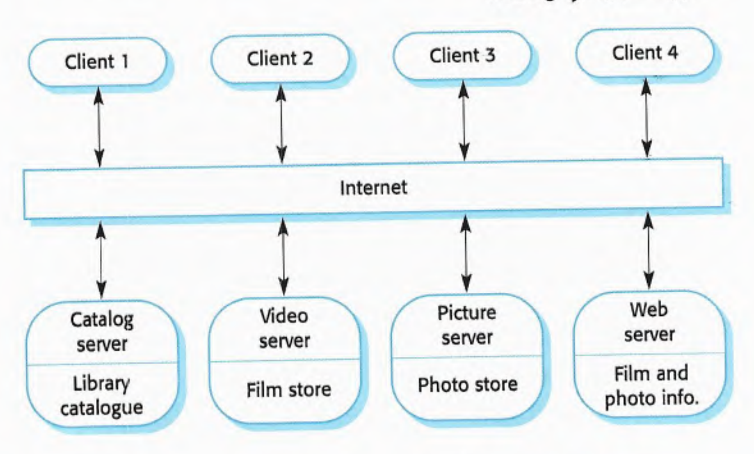
User interface
User interface management Authentication and authorization
Core business logic/application functionality System utilities
System support (OS, database etc.)

الشكل (35): نمط الطبقات Layered architecture



2 - نمط مخدم - عميل *client-server pattern*

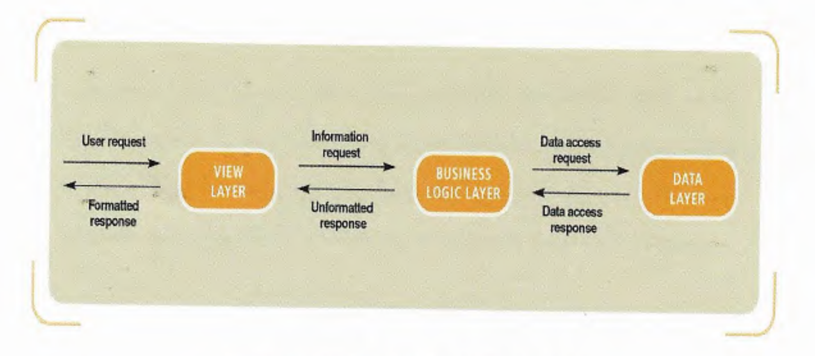
يبين الشكل *(36)* عن بنية مخدم – عميل وهو نموذج نظام موزع يوضح كيفية توزيع البيانات والمعالجة عبر مجموعة من المكونات موزعة فيزيائياً عبر أجهزة مختلفة. يتألف هذا النموذج من طبقتين أساسيتين: طبقة العميل الذي يطلب الموارد أو الخدمات من الخدم و طبقة الخدم الذي يدير موارد نظام المعلومات. يتواصل العميل والخدم عبر بروتوكولات محددة جيداً على الشبكة الفيزيائية. يمكن تواجد مجموعة من الخدمات التي تقدم خدمات محددة مثل الطباعة، وإدارة البيانات، وما إلى ذلك.



الشكل *(36)*: نمط مخدم - عميل *client-server architecture*

3 - نمط ثلاث طبقات مخدم – عميل *Three Layers Client / Server*

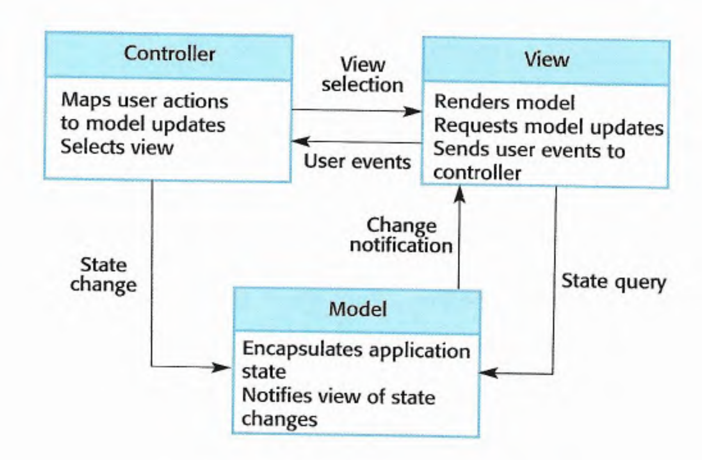
يبين الشكل *(37)* هذه البنية التي تعتبر بديل عن بنية مخدم -عميل التقليدية حيث يقسم برنامج التطبيق إلى عمليات مستقلة ويتكون من *3* طبقات: طبقة البيانات، طبقة منطق الأعمال، طبقة العرض ويتميز بمرونة إضافية وإمكانية الصيانة والموثوقية.



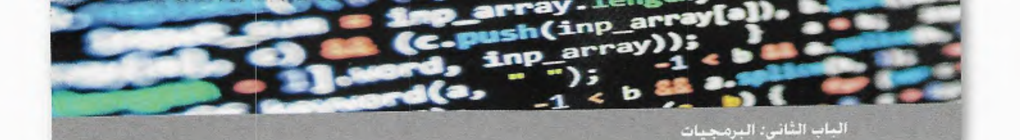
الشكل *(37)*: بنية ثلاث طبقات مخدم -عميل

4 - نمط نموذج-عرض- تحكم *Model-View-Controller (MVC) Pattern*

يبين الشكل *(38)* مثال حول هذا النمط:

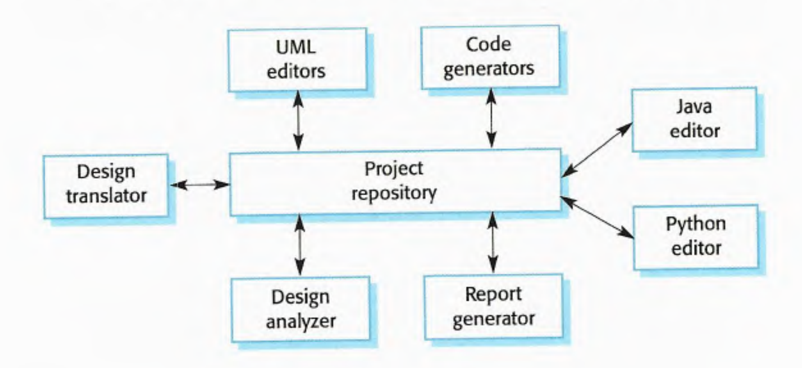


الشكل *(38)*: نمط نموذج – عرض – تحكم *(MVC)*



5 - نمط المستودع *Repository pattern*

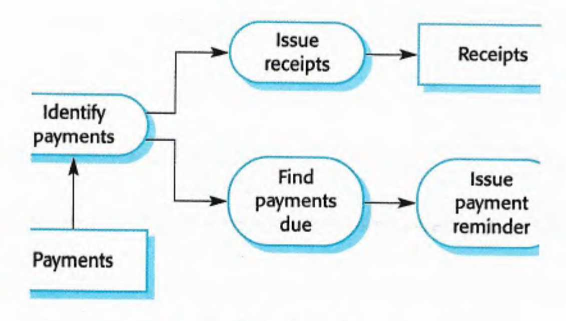
يستخدم هذا النموذج عند تبادل وتشارك النظم الفرعية نفس البيانات. ويمكن أن يتم ذلك عندما توجد البيانات المشتركة في قاعدة بيانات مركزية أو مستودع ويمكن الوصول إليها من قبل جميع الأنظمة الفرعية. يبين الشكل *(39)* مثال عن هذا النمط لبيئة عمل برمجية.



الشكل *(39)*: نمط المستودع

6 - نمط الأنابيب والفلترة *Pipe and filter pattern*

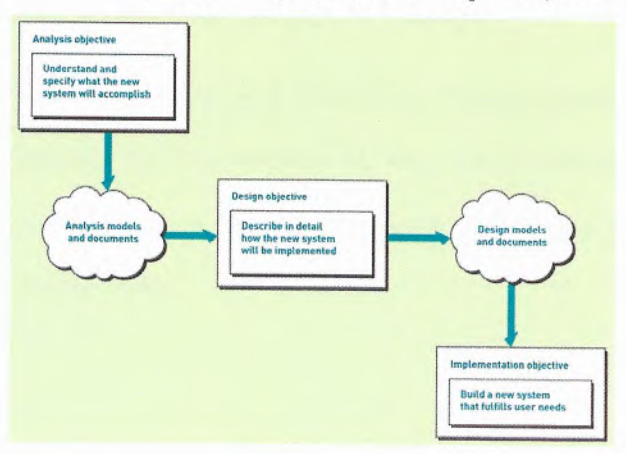
يستخدم هذا النموذج عندما تشكل مخرجات الأنظمة الفرعية دخلاً للأنظمة الفرعية الأخرى وتعالج مدخلاتها لنحصل على مخرجات نهائية. يكون هذا النموذج متسلسل يستخدم على نطاق واسع في نظم معالجة البيانات ولكنها غير مناسبة فعلياً للأنظمة التفاعلية. يبين الشكل *(40)* مثال عن هذا النمط.



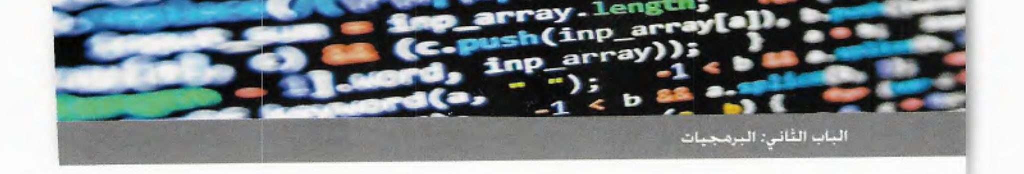
الشكل *(40)*: نمط الأنابيب والفلترة

2 - 4 - 4 - تصميم النظام التفصيلي *Detail System Design*

يتضمن التحديد التفصيلي لكيفية بناء النظام وبناء النماذج التصميمة التي يمكن أن تترجم باستخدام الأدوات واللغات البرمجية إلى النظام الفعلي كما هو موضح في الشكل *(41)*.

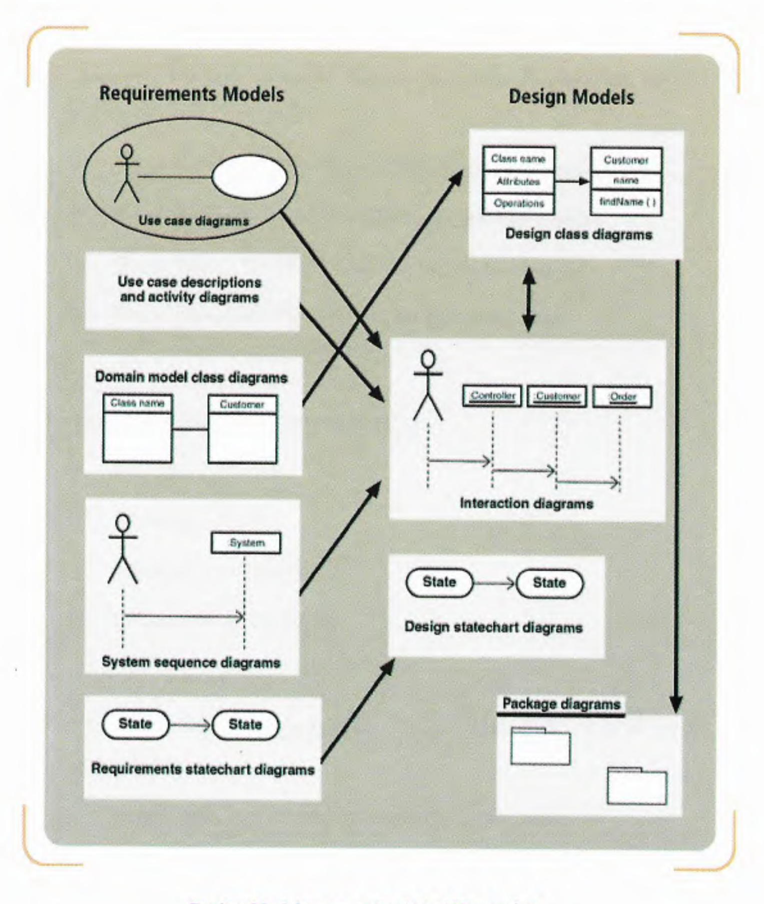


الشكل *(41)*: تصميم النظام



يعتبر بناء النماذج التصميمة من الأعمال الأساسية في مرحلة التصميم التفصيلي حيث يتم الاعتماد في ذلك على المخططات التركيبية والسلوكية المبدأية التي تم بناءها في المرحلة التحليلية كما هو مبين في الشكل *(42)* ومن ثم يتم المباشرة في بناء المخططات ذات الصلة بناء على حالات الاستخدام المحددة ومن ثم يتم تقسيم الكائنات إلى مجموعات لتصميم متعدد الطبقات وفق المراحل التالية:

- بناء المخططات التربيبة التفصيلية.
- تطوير مخططات تفاعلية لكل حالة أو سيناريو.
- تحديث المخططات التركيبية اعتماداً على ماستجد من تفاعلات للحصول على الخرج المراد بتزيد كل كائن وصف بأسماء الطرق *Methods* والسمات *Attributes* التي تخدم التفاعل المحدد لإنجاز الوظيفة المحددة.



الشكل (42): نماذج التصميم *Design Models*

2 - 4 - 5 - تمارين فصلية

1- يتضمن أسلوب غرضية التوجه في بناء البرمجيات *Object Oriented Development*

a. تحديد الدالات *functions* وعلاقات الربط فيما بينها.

b. تحديد الكائنات *objects* وعلاقات الربط فيما بينها.

c. تحديد الطرق *methods* وعلاقات الربط فيما بينها.

d. تحديد الوسائط *parameters* وطرق استدعائها.

2- تظهر *method signature* عادة في:

a. *Activity diagram.*

b. *Class diagram.*

c. *Sequence diagram.*

d. *Design class diagram.*

3- الطبقات الثلاث الموجودة في البنية المعمارية *three-layers-architecture:*

a. *view layer- business logic layer- data layer.*

b. *view layer- service layer- domain layer.*

c. *client layer- view layer- data layer.*

d. *client layer- domain layer- and server layer.*

2 - 5 - التنجيز والتسليم والاختبار

Software Implementation, Deployment and Testing

2 - 5 - 1 - التنجيز *System Implementation*

وهو تطوير البرمجيات أو تجميعها وفقاً للتصميم الذي تم إنشاؤه مسبقاً. ويعتبر التنجيز عملية ليست سهلة لأنه يتألف من العديد من الأنشطة المترابطة بما في ذلك تجهيز المكونات البرمجية والحصول على مكونات برمجية من مشاريع أخرى ودمج مكونات البرمجيات فيما بينها مع تأكيد تجانسها.

وتتضمن أنشطة التنجيز مايلي:

- اختيار لغة البرمجة.
- اختيار معيار التنجيز.
- بناء كود كل مكون من مكونات النظام.
- بناء وحدات قابلة للتنفيذ متوافق مع المعيار الذي تم اختياره.
- تثبيت برامج التطبيقات مع البنية التحتية لبرامج الأجهزة والبرامج الداعمة.
- اختبار البرمجيات من اختبارات الوحدة، بناء حالات الاختبار إلى إجراء اختبارات التكامل.

ويمكن ضبط ترتيب عمليات التنجيز بأحد الأشكال التالية:

- المدخلات والعملية والانتاج *Input, Process, Output (IPO)*: وهو الترتيب الذي ينفذ أولاً وحدات الإدخال ثم الوحدات الإجرائية، وأخيراً وحدات الإنتاج.
- عملية التطوير من الأعلى إلى الأسفل *Top-down*: وهو الترتيب الذي ينفذ وحدات المستوى الأعلى أولاً.
- عملية التطوير من الأسفل إلى الأعلى *Bottom-up*: وهو الترتيب الذي ينفذ وحدات مفصلة على مستوى منخفض أولاً.

2 - 5 - 2 - النشر Deployment

هو مجموعة من الأنشطة المطلوبة لتجهيز النظام البرمجي وتسليمه للزبون حسب الشكل *(43)* والتي تتضمن مايلي:

Core processes	Iterations 1	2	3	4	5	6
Identify the problem and obtain approval.						
Plan and monitor the project.						
Discover and understand details.						
Design system components.						
Build, test, and integrate system components.						
Complete system tests and deploy the solution.						

Deployment activities
Perform system and stress tests.
Perform user acceptance tests.
Convert existing data.
Build training materials and conduct training.
Configure and set up production environment.
Deploy the solution.

الشكل *(43)*: النشر *Deployment*

1- **تشغيل اختبارات النظام، اختبارات الإجهاد**، واختبارات قبول المستخدم.

2- **تحويل البيانات وتهيئتها:** وذلك يتضمن:

- إعادة استخدام قواعد البيانات الحالية من خلال تعديل البيانات

الحالية أو تحديثها.

- إعادة تحميل قواعد البيانات من خلال نسخ البيانات وتحويلها، تصدير واستيراد البيانات من نظام إدارة قواعد البيانات المميز وإدخال البيانات من الوثائق الورقية.

3- **تدريب المستخدمين:**

- يجب توفير التدريب للمستخدمين النهائيين ومشغلي النظم.
- من الضروري أن يركز التدريب للمستخدمين النهائيين على الاستخدام العملي لعمليات أو وظائف تجارية معينة، مثل إدخال الطلبات أو مراقبة المخزون أو المحاسبة.
- يمكن أن يكون تدريب مشغل النظام أقل رسمية عندما يكون المشغلون ليسوا مستخدمين نهائيين، حيث يمكن لمشغلي الكمبيوتر ذوي الخبرة والإداريين تعلم معظم أو كل ما يحتاجون إلى معرفته من خلال الدراسة الذاتية بالإضافة لإعادة تحميل قواعد البيانات.

4- **التوثيق:**

- وثائق النظام.
- وثائق البرنامج.
- وثائق المستخدم.

5- **نشر الحل:**

الأمور التي يجب مراعاتها عند النشر:

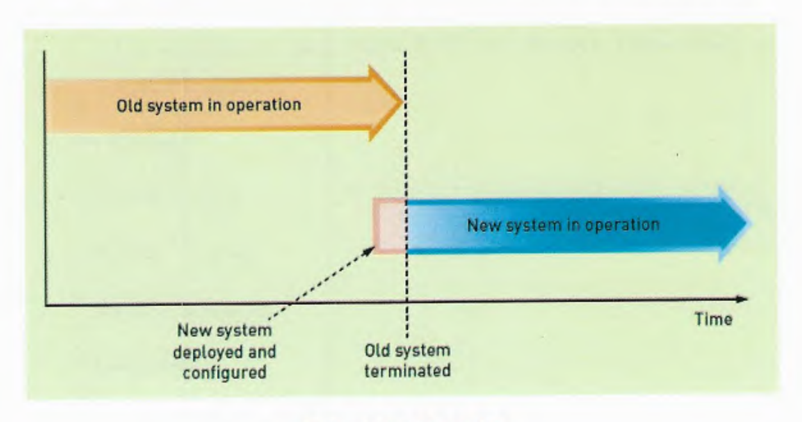
- تحمل تكاليف تشغيل النظامين بالتوازي.
- الكشف عن الأخطاء وتصحيحها في النظام الجديد.

- احتمال تعطيل الشركة وعملياتها أثناء عملية النشر.

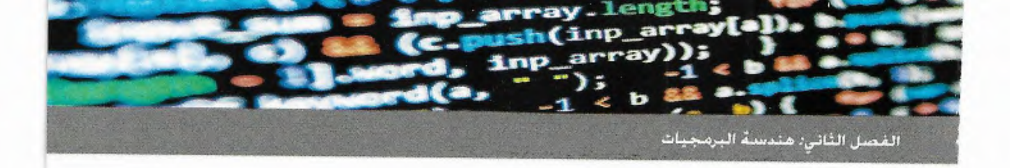
ويمكن اعتماد أحد الأشكال التالية عند نشر المنتج:

1 - النشر المباشر *Direct Deployment*

وهي الطريقة التي تقوم بتثبيت نظام جديد يجعلها تعمل بسرعة، ويجب على الفور إيقاف أي أنظمة متداخلة. المزايا الأساسية لهذه الطريقة تكمن في بساطتها وانخفاض التكلفة. وبما أن النظامين القديم والجديد لا يتم تشغيلهما بالتوازي، فإن هناك عدداً أقل من المسائل اللوجستية التي يجب إدارتها. العيب الرئيسي هو المخاطر العالية لأن الأنظمة القديمة لا تعمل بالتوازي معها وبالتالي لا يوجد نسخ احتياطي في حالة فشل النظام الجديد. ويبين الشكل *(44)* طريقة النشر المباشر.

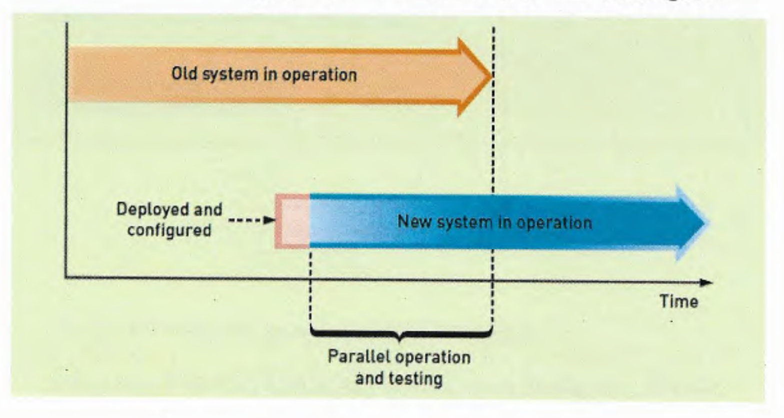


الشكل *(44)*: النشر المباشر



2 - النشر المتوازي *Parallel Deployment*

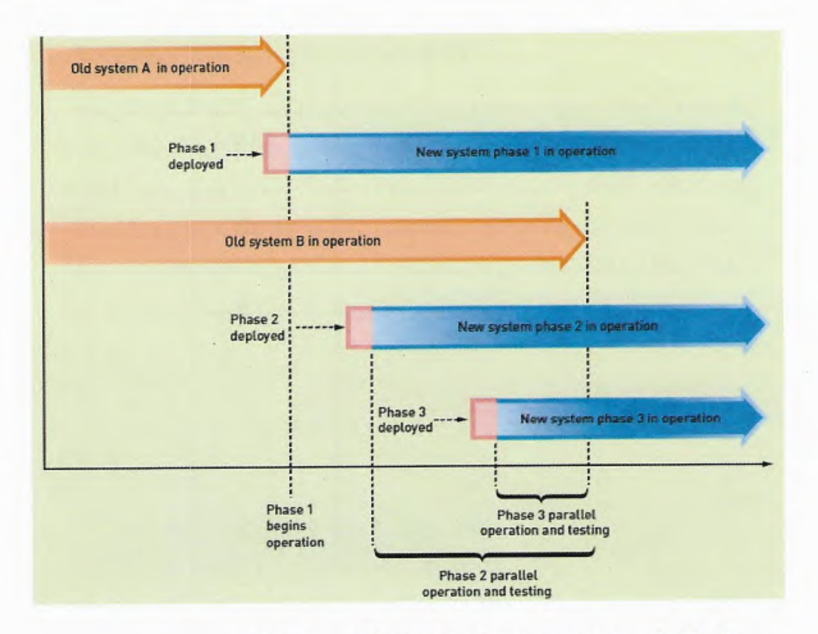
وهي طريقة نشر تعمل بالنظام القديم والجديد لفترة زمنية محددة وتمتاز بانخفاض المخاطر وارتفاع التكلفة. تستمر الأنظمة القديمة في العمل حتى يتم اختبار النظام الجديد بدقة وتقرر أن تكون خالية من الأخطاء وجاهزة للعمل بشكل مستقل. غالبا ما يتم تحديد الوقت المخصص للعملية الموازية مسبقاً ويقتصر على تقليل تكلفة التشغيل المزدوج. ويبين الشكل *(45)* طريقة النشر المتوازي.



الشكل *(45)*: النشر المتوازي

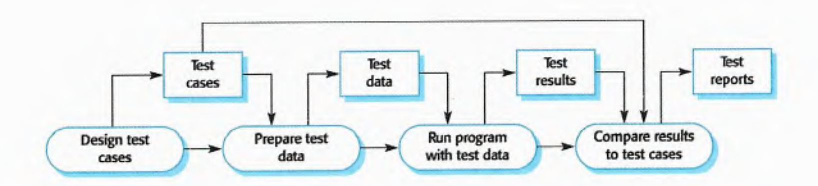
3 - النشر المرحلي *Phased Deployment*

طريقة نشر تقوم بتثبيت نظام جديد وتشغيله في سلسلة من الخطوات أو المراحل وتضيف كل مرحلة مكونات أو وظائف إلى نظام التشغيل. خلال كل مرحلة يتم اختبار النظام للتأكد من أن النظام جاهز للمرحلة التالية وتستمر الأنظمة القديمة في العمل حتى يتم اختبار النظام الجديد بدقة وتقرر أن تكون خالية. ويبين الشكل *(46)* طريقة النشر المرحلي.



2 - 5 - 3 - اختبار البرمجيات *Software Testing*

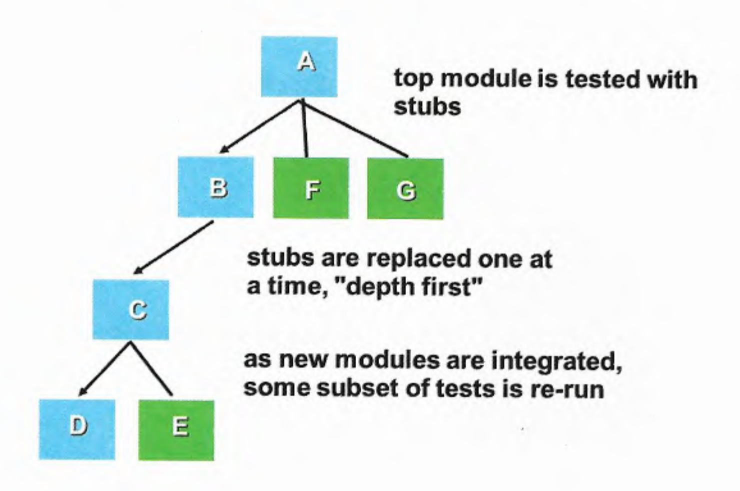
وهي عملية تشغيل النظام تحت الاختبار بغرض العثور على الأخطاء قبل التسليم إلى المستخدم النهائي. يبين الشكل *(47)* مخطط عملية الاختبار *Testing process*

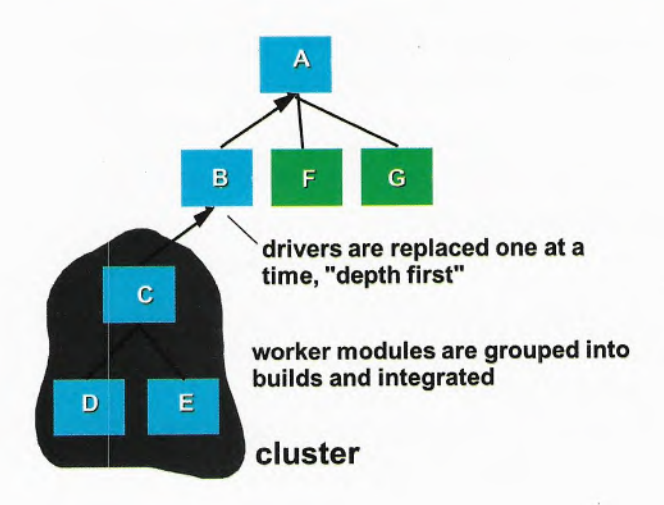


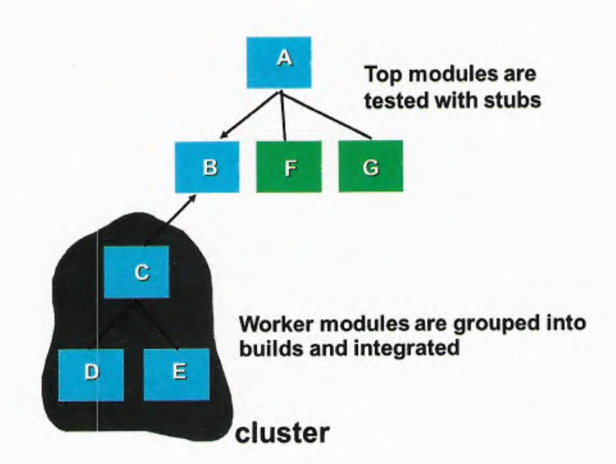
1 - **اختبار التطوير** ***Development Testing:*** حيث يتم اختبار النظام أثناء التطوير لاكتشاف الأخطاء والعيوب. تقسم اختبارات التطوير بشكل تزايدي إلى عدة أنواع:

- **اختبار الوحدات** ***Unit Testing:*** وهي عملية اختبار كائنات النظام بشكل فردي ومعزول. يمكن أن تكون هذه المكونات وظائف أو طرق فردية داخل الكائن المختبر.

- **اختبار التكامل** ***Integration Testing:*** وهي عملية اختبار كائنات النظام بشكل تفاعلي فيما بينها ولها عدة أشكال حيث يبين الشكل *(48)* طريقة الاختبار من الأعلى إلى الأسفل *Top Down* والشكل *(49)* طريقة الاختبار من الأسفل إلى الأعلى *Bottom Up* والشكل *(50)* اختبار السندويش *Sandwich Testing*.




A
B
F
G
C
D
E
drivers are replaced one at a time, "depth first"
worker modules are grouped into builds and integrated
cluster


A
Top modules are tested with stubs
B
F
G
C
D
E
Worker modules are grouped into builds and integrated
cluster

التطوير دمج المكونات لإنشاء نسخة من النظام ومن ثم اختبار النظام المتكامل ككل.

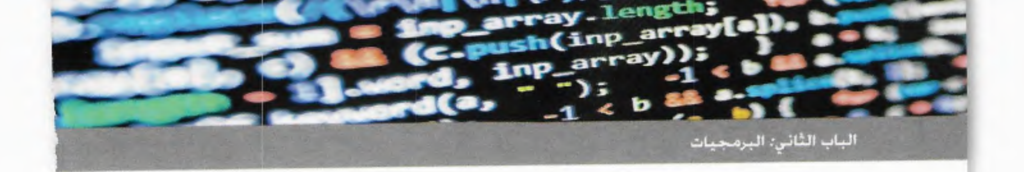
- **اختبار الانحدار (الارتداد) *Regression Testing*:** وهو إعادة تنفيذ بعض المجموعات الفرعية من الاختبارات التي أجريت بالفعل للتأكد من أن التغييرات لم تحدث آثار جانبية غير مقصودة.

2 - **اختبار الإصدار *Release Testing*:** هو عملية اختبار إصدار معين من نظام مخصص للاستخدام خارج فريق التطوير حيث يقوم فريق اختبار منفصل باختبار نسخة كاملة من النظام قبل أن يتم طرحها للمستخدمين .

3 - **اختبار المستخدم *User Testing*:** هو مرحلة في عملية الاختبار يقدم فيها المستخدمين أو العملاء المدخلات والمشورة بشأن اختبار النظام. ولاختبار القبول نوعين:

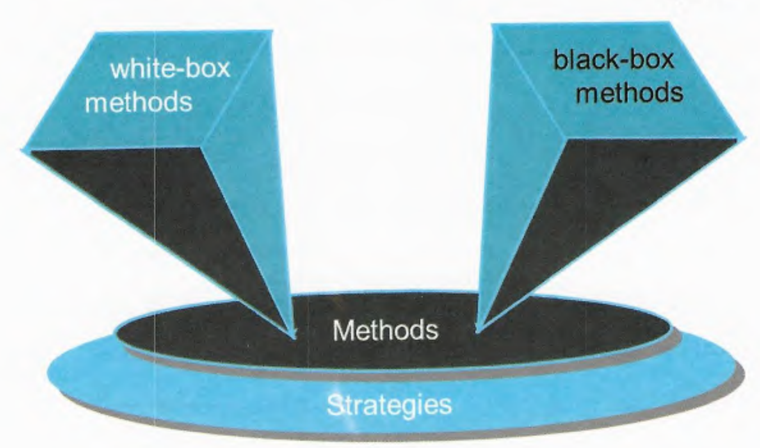
- **اختبار ألفا *Alpha testing*:** وهو اختبار في موقع المطورين لتحديد ما إذا كان النظام جاهزاً للمستخدم وينبغي أن يستند ذلك إلى معايير قبول محددة بوضوح حيث يعمل مستخدمي البرنامج مع فريق التطوير لاختبار البرنامج في موقع المطور.

- **اختبار بيتا *Beta testing*:** يختبر العملاء النظام لتحديد ما إذا كان مستعداً لقبولها من مطوري النظام ونشرها في بيئة العميل أم لا. يتم توفير إصدار من البرنامج للمستخدمين للسماح لهم للتجربة ورفع المشاكل التي يكتشفونها إلى مطوري النظام.



2 - 5 - 4 - مناظير اختبار البرمجيات *Software Testing Views*

يوضح الشكل *(51)* منظورين أساسيين يتم إجراء الاختبارات البرمجية وفقهما وهما اختبارات الصندوق الأبيض واختبارات الصندوق الأسود.



الشكل *(51)*: مناظير الاختبارات البرمجية

1 - اختبارات الصندوق الأبيض *White Box Testing*

وتسمى أيضاً اختبار الصندوق الزجاجي حيث يتم توليد الاختبارات على أساس الكود البرمجي مع الأخذ بعين الاعتبار الآلية الداخلية للنظام أو مكون منه. يتم من خلالها اختبار النظام بتفصيل كبير ولكن مع تكلفة قد تكون عالية لذلك تعتبر هذه الاختبارات فعالة في اختبار سلوك النظام والتأكد من صحة مخرجاته ولكنها قد لا تستطيع الكشف عن السلوك المفقود من النظام. من أهم مراحل إجراء اختبارات الصندوق الأبيض مايلي:

- تحديد المكون تحت الاختبار.

- نمذجة سلوك المكون بمخططات تدفق مثل (*Control Flow Graph*
CFG).

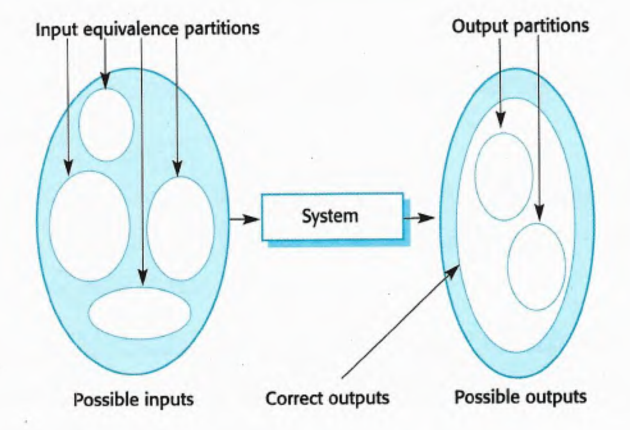
- استخدام معيار تغطية مناسب *Coverage criteria* للتأكد أن هيكلية المكون تحت الاختبار قد تم تغطيتها وتنفيذها بشكل مرضي أو كامل أثناء الاختبار.

من معايير التغطية المستخدمة في اختبارات الصندوق الأبيض تغطية العبارة *Statement coverage*، تغطية الفرع *Branch coverage* ، تغطية المسار *Path coverage*.

2 - اختبارات الصندوق الأسود *Black Box Testing*

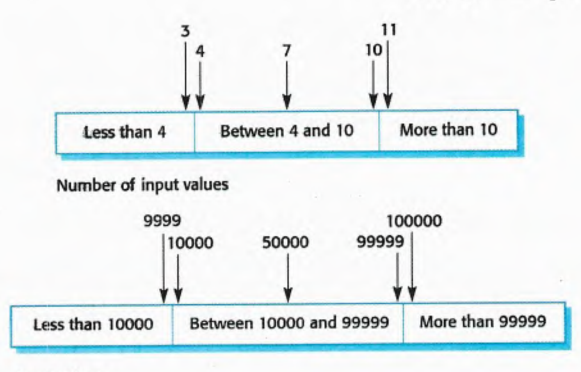
تختبر النظام ك "صندوق الأسود" بغض النظر عن بنيته الداخلية على أساس المواصفات أو المتطلبات حيث يركز فقط على المخرجات استجابة للمدخلات المختارة وظروف التنفيذ. يتم اعتماد هذه الاختبارات لتقييم مدى التزام النظام أو المكون بمتطلبات وظيفية محددة وتعتبر اختبارات فعالة في اختبار السلوك المطلوب ولكنها ضعيفة في الكشف عن أي سلوك إضافي.

من أهم المبادئ المعتمدة في اختبارات الصندوق الأسود هو مبدأ التقسيم المتكافئ *Equivalence Partitioning* المبين في الشكل *(52)* حيث يقسم مجال الدخل إلى مجموعات من البيانات المتكافئة للمدخلات التي تنتج أساساً نفس النتائج وبالتالي يكتفى بممثل أو أكثر عن كل مجموعة مولدة لتوليد حالات الاختبار.



الشكل *(52)*: التقسيم المكافئ *Equivalence Partitioning*

ومن أحد الطرق المستخدمة في اختيار عينات الاختبار من كل مجموعة هو تحليل حدود كل مجموعة *Boundary Analysis* كما هة مبين في الشكل *(53)*:



الشكل *(53)*: تحليل قيمة الحدود *Boundary Value Analysis*

من أهم الاختبارت التي تندرج تحت منظور اختبارات الصندوق الأسود

- **اختبار السيناريو** ***(Scenario-based)***: يصف نمط اختبار السيناريو أسلوب عمل النظام من وجهة نظر المستخدم وهو ينطوي على فحص كل شرط ووضع اختبار أو اختبارات لذلك. يشير الفشل في هذا المستوى إلى فشل النظام في تلبية متطلبات مرئية للمستخدم.
- **اختبار الإجهاد** ***Stress Testing***: في اختبار الإجهاد نحن 'نحمّل' النظام بحمولة أكبر بكثير من المعتاد ومن ثم ندرس أداؤه.
- **اختبار الأمان** ***Security Testing***: يتم التحقق من أن آليات الحماية المضمنة في النظام سوف تحميها من الاختراق.
- **اختبار الأداء** ***Performance Testing***: وفيه يختبر أداء التشغيل وزمن استجابته في سياق نظام متكامل.
- **الاختبار القائم على النموذج** ***Model-based Testing***: وفيه يتم اختبار سلوك النظام بناء على نموذج محدد.

2 - 5 - 5 - تمارين فصلية

1- ترتكز عملية الاختبار البرمجية *Software testing* على فحص:

a. الكود *Code*.

b. التصميم *Design*.

c. التحليل *Analysis*.

d. الوثائق *Documentation*.

2- من أهداف عمليات اختبار الوحدات *unit testing* فحص:

a. استقرار الكود البرمجي.

b. جودة أداء خوارزميات الحل البرمجي.

c. أصغر وحدة برمجية مكتوبة للتأكد من صحة عملها.

d. القدرة على تجاوز الخطأ.

3- من ميزات استخدام تقنية الاختبار التكاملي من الأسفل إلى الأعلى *Bottom-up Integration testing*:

a. اختبار نقط اتخاذ القرار بشكل مبكر.

b. عدم الحاجة لاستخدام *drivers* لإجراء الاختبار.

c. عدم الحاجة لاستخدام *stubs* لإجراء الاختبار.

d. الإجابات أعلاه جميعها غير صحيحة.

4- تعتمد اختبارات *condition testing* على:

a. إيجاد كل المسارات *path testing*.

b. اختبار جميع الفروعات *branches*.

c. تحديد المسارات الواصلة بين تعريف واستخدام المتحولات.

d. اختبار البنية الداخلية للحلقات.

5- حسب نموذج التطوير البرمجي *SDLC* يمكن البدء بالاختبارات البرمجية في طور:

a. التحليل *Analysis*.

b. التصميم *Design*.

c. التنجيز *Implementation*.

d. النشر *Deployment*.

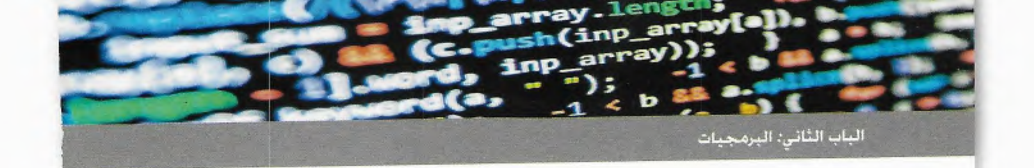
6- من المستبعد أن يقوم ______ بعمليات الاختبار *software testing* .

a. *Developer*.

b. *Tester*.

c. *Outsourcer*.

d. *Project manager*.



الباب الثالث

الذكاء الصنعي

(Artificial Intelligence)

الفصل الأول

حساب الفرضيات

The Propositional Calculus

1 - 1 - أهمية حساب الفرضيات

- تُشكّل الفرضيات ثنائية القيمة وصفاً للعالم (ما هو صحيح في هذا العالم وما هو غير صحيح).
- أمثلة:
  - "لا توجد الكتلة A على الأرض".
  - "توجد الكتلة A إما فوق الكتلة B وإما فوق الكتلة C".
- يُمكن صياغة بعض المعلومات عن العالم على شكل قيود على قيم الفرضيات فيه.
- تُمثّل هذه القيود معارف مهمة حول العالم.
- كما يُمكن استخدامها لتوليد قيم فرضيات أخرى غير قابلة للقياس مباشرة.

مثال:

- ليكن لدينا ربوط قادر على حمل كتلة. إذا كانت هذه الكتلة قابلة للحمل (أي غير ثقيلة جداً) وإذا كانت شحنة بطارية الربوط كافية.
- إذا تحقق كل من هذين الشرطين سيقوم الربوط برفع الكتلة التي يُمسكها وذلك بتحريك ذراعه.

- يُمكن تمثيل مختلف هذه الشروط باستخدام الفرضيات ثنائية القيمة التالية:

  - *BAT_OK (البطارية مشحونة)*
  - *LIFTABLE (الكتلة قابلة للحمل)*
  - *MOVES (الذراع تتحرك)*

- لنفترض أن الربوط يتحسس قيمة *BAT_OK* عن طريق قراءة حساس، وقيمة *MOVES* بوساطة حساسات موضعه عليه. إلا أنه لا يتحسس قيمة *LIFTABLE* والتي تُشكّل قيمة مهمة له.

- **لنجري الآن المحاكمة التالية:**

- نعرف أنه إذا كان لكل من *BAT_OK* و *LIFTABLE* القيمة *1* فسيكون لـ *MOVES* القيمة *1* أيضاً.

- كذلك إذا كان لـ *MOVES* القيمة *0* عندما يحاول الربوط أن يحرك الكتلة فإننا نعرف أنه إما لـ *BAT_OK* وإما لـ *LIFTABLE* (أو لكليهما) القيمة *0*.

- إلا أنه إذا كان لـ *BAT_OK* القيمة *1* (حسب الحساس المرتبط) فيبقى أن يكون لـ *LIFTABLE* القيمة *0*.

- **بما أننا نستطيع المحاكمة هكذا، لنجعل الربوط يحاكم مثلنا!**

- نحتاج إذاً للغة لنمذجة العالم.

- كما نحتاج إلى آلية استدلال نستطيع بوساطتها تحقيق المحاكمة المطلوبة.

- يُقدم حساب الفرضيات والمتفرع عن الجبر المنطقي الأدوات الضرورية لذلك.

1 - 2 - الشكل Syntax - مكونات اللغة

- الذرات *Atoms*

- لدينا أولاً الذرتان *T, F*.

- المجموعة المعدودة وغير المنتهية من السلاسل الحرفية التي تبدأ بحرف كبير. مثلاً:

P, Q, R, P1, P2, ON_A_B,...

- الروابط *Connectors*

- $\vee$ والتي تُدعى "أو" *"or"*

- $\wedge$ والتي تُدعى "و" *"and"*

- $\neg$ والتي تُدعى "لا" *"not"*

- $\Rightarrow$ والتي تُدعى "يقتضي" *"implies"*

- الصيغ جيدة التركيب *Well-Formed Formulas WFF*

- أي ذرة هي صيغة جيدة التركيب. مثلاً: *R , P*

- إذا كانت *w1, w2* صيغاً جيدة التركيب فإن كل من الصيغ التالية هي صيغة جيدة التركيب:

- $w1 \vee w2$

- $w1 \wedge w2$

- $w1 \Rightarrow w2$

- $\neg w1$

ندعو الذرة أو الذرة المسبوقة بإشارة النفي $\neg$ بحرفي *Literal*.

ندعو $w1$ في $w1 \Rightarrow w2$ بمقدمة الاقتضاء *Antecedent* و $w2$ بنتيجة الاقتضاء *Consequent*.

- لا يوجد صيغ أخرى جيدة التركيب. فمثلاً $\neg\neg \Rightarrow P$ ليست صيغة *wff*.

أمثلة:

- $(P \wedge Q) \Rightarrow \neg P$
- $P \Rightarrow \neg P$
- $P \vee P \Rightarrow P$
- $(P \Rightarrow Q) \Rightarrow (\neg Q \Rightarrow \neg P)$
- $\neg\neg P$

1 - 3 - الدلالة *Semantic*

إذا أُعطيت قيم الذرات في تفسير *interpretation* ما، فيُمكن استخدام جدول الحقيقة لحساب قيمة أي صيغة *wff* في هذا التفسير. يُعطي جدول الحقيقة دلالة (معنى) الروابط في حساب الفرضيات.

$w1$	$w2$	$w1 \wedge w2$	$w1 \vee w2$	$\neg w1$	$w1 \Rightarrow w2$
True	*True*	*True*	*True*	*False*	*True*
True	*False*	*False*	*True*	*False*	*False*
False	*True*	*False*	*True*	*True*	*True*
False	*False*	*False*	*False*	*True*	*True*

1 - 4 - مفهوم قابلية التحقيق *Satisfiability* والنماذج *Models*

- نقول عن تفسير إنه يُحقق صيغة *wff* إذا كان للصيغة القيمة *True* تحت هذا التفسير.
- نقول عن تفسير يُحقق صيغة إنه نموذج لها *Model*.
- نقول عن صيغة أنها غير قابلة للتحقيق *Inconsistent* أو *Unsatisfiable* إذا لم يوجد تفسير يُحققها.
- نقول عن صيغة إنها صالحة *Valid* إذا كان لها القيمة *True* من أجل كل تفسير لذراتها المكونة.

أمثلة:

- الصيغ: **False** و $P\wedge\neg P$ غير قابلة للتحقيق.
- مجموعة الصيغ: $\{P\vee Q, P\vee\neg Q, \neg P\vee Q, \neg P\vee\neg Q\}$ غير قابلة للتحقيق، إذ لا يوجد أي تفسير يجعل جميع صيغ هذه المجموعة صحيحة. *(استخدم جدول الحقيقة للتأكد من ذلك)*.
- جميع الصيغ التالية صالحة:
  - $P\Rightarrow P$
  - T
  - $\neg(P\wedge\neg P)$
  - $Q\vee T$
  - $[(P\Rightarrow Q)\Rightarrow P]\Rightarrow P$
  - $P\Rightarrow(Q\Rightarrow P)$

لاحظ أن استخدام جدول الحقيقة للتأكد من صلاحية صيغة يتطلب تعقيداً أسياً وفق عدد الذرات المكونة لها، إذ يجب إيجاد قيمة الصيغة من أجل جميع القيم الممكنة لذراتها.

1 - 5 - التكافؤ Equivalence

نقول عن صيغتين *wff* إنهما متكافئتان إذا وفقط إذا كانت لهما قيم الحقيقة نفسها من أجل كل التفاسير. (سنرمز للتكافؤ بالإشارة ≡).

يُمكن استخدام جدول الحقيقة لبرهان التكافؤات التالية:

- قوانين دومرغان

$$\neg (w1 \vee w2) \equiv \neg w1 \wedge \neg w2$$

$$\neg (w1 \wedge w2) \equiv \neg w1 \vee \neg w2$$

- قانون عكس الإيجاب

$$(w1 \Rightarrow w2) \equiv (\neg w2 \Rightarrow \neg w1)$$

إذا كانت $w1$ و $w2$ متكافئتين فإن الصيغة التالية صالحة:

$$(w1 \Rightarrow w2) \wedge (w2 \Rightarrow w1)$$

وبسبب هذه الحقيقة فإن المفهوم:

$$w1 \equiv w2$$

كثيراً ما يُستخدم كاختصار لـ

$$(w1 \Rightarrow w2) \wedge (w2 \Rightarrow w1)$$

1 - 6 - قواعد الاستدلال *Rules of Inference*

يوجد عدة طرائق لتوليد صيغ *wff* بدءاً من صيغ أخرى، ندعو هذه العملية بالاستدلال *Inference* أو الاستنتاج. يكون لقاعدة الاستدلال الشكل:

γ يمكن أن تستنتج من α

نعطي فيما يلي مجموعة من قواعد الاستدلال المستخدمة في حساب الفرضيات:

- **يمكن للصيغة *w2* أن تستنتج من الصيغتين *w1* و *w1⇒w2*** *(Modus Ponens)*
- **يمكن للصيغة *w1∧w2* أن تستنتج من الصيغتين *w1* و *w2*** *(إدخال العطف ∧)*
- **يمكن للصيغة *w2∧w1* أن تستنتج من *w1∧w2*** *(العطف تبديلي)*
- **يمكن للصيغة *w1* أن تستنتج من الصيغة *w1∧w2*** *(حذف العطف ∧)*
- **يمكن للصيغة *w1∨w2* أن تستنتج إما من *w1* وإما من *w2*** *(إدخال الفصل ∨)*
- **يمكن للصيغة *w1* أن تستنتج من الصيغة *(¬w1) ¬*** *(حذف النفي ¬)*

1 - 7 - البرهان *Proof*

تُدعى سلسلة الصيغ جيدة التراكيب $\{w_1,w_2,......w_n\}$ ببرهان أو استنتاج w_n من مجموعة من الصيغ جيدة التركيب Δ إذا وفقط إذا كانت:

كل صيغة w_i في السلسلة هي إما موجودة في Δ وإما يمكن استنتاجها من صيغة أو (عدة صيغ) سابقة في السلسلة باستخدام إحدى قواعد الاستنتاج.

إذا وجد برهان لـ w_n من Δ فإننا نقول إن w_n هي نظرية *Theorem* للمجموعة Δ ونكتب:

$$\Delta \mid\!- wn$$

مثال: ليكن لدينا مثلاً مجموعة الصيغ:

$$\Delta = \{P, R, P \Rightarrow Q\}$$

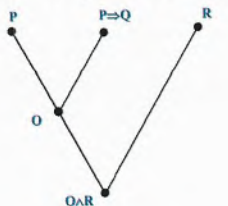
فإن السلسلة التالية هي برهان لـ $Q\grave{U}R$ باستخدام قواعد الاستدلال المعطاة في العبارة السابقة:

$$\{P, P \Rightarrow Q, Q, R, Q \wedge R\}$$

يُمكن تمثيل سلسلة البرهان باستخدام شجرة.

تُعنون كل عقدة في شجرة البرهان بصيغة *wff*.

تكون هذه الصيغة إحدى صيغ Δ أو صيغة مستنتجة من آبائها في الشجرة باستخدام قواعد الاستنتاج. وتكون الشجرة برهاناً لصيغة جذر الشجرة.



1 - 8 - الاستتباع *Entailment*

إذا كان لصيغة *wff* القيمة *True* من أجل كل التفسيرات التي تجعل قيمة كل صيغة في مجموعة Δ من الصيغ *True*. فإننا نقول إن Δ تستتبع *Entail* منطقياً *w*. وإن *w* تتبع *Follow* منطقياً Δ أو إن *w* هي نتيجة منطقية *Logical Consequence* من Δ.

نستخدم الرمز |= للدلالة على الاستتباع المنطقي ونكتب $D|=w$.

أمثلة:

- $P\} \models P\}$
- $\{P, P\Rightarrow Q\} \models Q$
- $F \supset w$ (أي صيغة :*w*)
- $P\wedge Q \models P$

مثال: لنفرض مثلاً، في مثال الربوط السابق، لدينا *BAT_OK* صحيحة (البطارية مشحونة) و *MOVES*¬ صحيحة (الذراع لا تتحرك). وأننا نُمثّل بعض معرفتنا حول العالم بالصيغة:

$$BAT_OK \wedge LIFTABLE \Rightarrow MOVES$$

وبهذا فإن لدينا ثلاث صيغ: اثنتان منها تصفان حالة معينة للعالم والثالثة تصف معرفة عامة حول العالم.

يُمكن استخدام جدول الحقيقة لإظهار أن *LIFTABLE*¬ تتبع منطقياً لهذه الصيغ الثلاث.

BAT_OK	$LIFTABLE$	$MOVES$	$\neg MOVES$	$BAT_OK \wedge LIFTABLE \Rightarrow MOVES$	$\neg LIFTABLE$
F	F	F	T	T	T
F	F	T	F	T	T
F	T	F	T	T	F
F	T	T	F	T	F
T	F	F	T	T	T
T	F	T	F	T	T
T	T	F	T	F	F
T	T	T	F	T	F

بما أن، باستخدام الاستتباع المنطقي، للصيغة $\neg LIFTABLE$ القيمة *True* في جميع التفسيرات التي تكون فيها هذه الصيغ الثلاث صحيحة. فإنها يجب بالتأكيد أن تأخذ القيمة *True* في تفسيرنا المعتمد للعالم.

وبهذا فإن الفرضية (والتي هي جزء من تفسيرنا المعتمد) "الكتلة غير قابلة للحمل" يجب أن تكون صحيحة.

إن استبدال طرائق جدول الحقيقة بطرائق البرهان يعطي كلفة حسابية أقل بكثير في حساب الفرضيات وحساب الإسناديات.

1 - 9 - قاعدة جديدة للاستدلال: الحل Resolution

تُعطى قاعدة الحل في حساب الفرضيات كما يلي:

لتكن Σ_1 و Σ_2 مجموعتين من الحرفيات (عبارتين) و λ ذرة.
من: $\{\lambda\} \cup \Sigma_1$ و $\{\neg\lambda\} \cup \Sigma_2$ يمكن أن نستنتج: $\Sigma_1 \cup \Sigma_2$
والتي تدعى بناتج الحل Resolvent (أو اختصاراً الحل) للعبارتين Σ_1

و Σ_2.

كما تدعى الذرة λ بالحَل، والإجراء بالحل.

مثال: يؤدي حل $R \vee P$ و $\neg P \vee Q$ إلى $R \vee Q$

يُمكن أن نكتب العبارات التي تمّ حلها كاقتضاءات $\neg R \Rightarrow P$ و $P \Rightarrow Q$.

يؤدي تطبيق قاعدة الاستدلال والمدعوة بالتسلسل Chaining على الاقتضائين السابقين إلى $\neg R \Rightarrow Q$ والتي تُكافئ $R \vee Q$. سنرى لاحقاً أن التسلسل هو حالة خاصة من الحل.

مثال: يؤدي حل R و $\neg R \vee P$ إلى P

وبما أن العبارة الثانية تُكافئ $R \Rightarrow P$ فنلاحظ أن مودس بوننس هي أيضاً حالة خاصة من الحل.

مثال: يؤدي حل $P \vee Q \vee R \vee S$ مع: $\neg P \vee Q \vee W$ على P إلى $Q \vee R \vee S \vee W$

لاحظ أن Q تظهر مرة واحدة في عبارة الحل (والتي هي مجموعة).

مثال: يؤدي حل $P \vee Q \vee \neg R$ مع $P \vee W \vee \neg Q \vee R$ على Q إلى $P \vee \neg R \vee R \vee W$

ويؤدي حلهما على R إلى $P \vee Q \vee \neg Q \vee W$

في هذه الحالة. وبما أن لكل من $\neg R \vee R$ و $Q \vee \neg Q$ القيمة *True*، فإن قيمة كل من الحلين *True*.

لاحظ أنه في هذا المثال يجب الحل إما على Q وإما على R وليس على كليهما بالوقت نفسه. أي أن $P \vee W$ ليس حلاً للفقرتين.

يؤدي حل λ (حرفياً موجباً) مع $\neg\lambda$ (حرفياً سالباً) إلى العبارة الخالية. أي من λ و $\neg\lambda$ يُمكن استنتاج F لأن λ و $\neg\lambda$ متناقضتان.

تكون أي مجموعة من الصيغ *wffs* و التي تحوي λ و $\neg\lambda$ مجموعة غير قابلة للتحقيق *Unsatisfiable*.

وبالمقابل فإن أي عبارة تحوي ذرة ونفيها (مثل λ و $\neg\lambda$) يكون لها القيمة *True* بغض النظر عن قيمة λ.

1 - 10 - تحويل الصيغ *wffs* إلى عطف عبارات

يمكن تحويل أي صيغة في حساب الفرضيات إلى عطف عبارات مكافئ.

ونقول عن الصيغة المكتوبة كعطف عبارات إنها في شكل العطف النظامي: *Conjunctive Normal Form(CNF)*

نستعرض عبر المثال التالي خطوات تحويل صيغة إلى الشكل *CNF*. لتكن الصيغة المطلوب تحويلها:

$$\neg(P \Rightarrow Q) \vee (R \Rightarrow P)$$

1. حذف إشارات الاقتضاء وذلك باستخدام الشكل المكافئ مع $\vee$:

$$\neg(\neg P \vee Q) \vee (\neg R \vee P)$$

2. خفض مجال النفي ¬ باستخدام قوانين دومرغان وبحذف إشارتي النفي المتتاليتين ¬¬:

$$(P\wedge\neg Q)\vee(\neg R\vee P)$$

3. استخدام قوانين التجميع والتوزيع للوصول إلى الشكل CNF:

$$(P\vee\neg R\vee P)\wedge(\neg Q\vee\neg R\vee P)$$

ثم:

$$(P\vee\neg R)\wedge(\neg Q\vee\neg R\vee P)$$

نقوم عادةً بالتعبير عن عطف عبارات *(الشكل النظامي)* بمجموعة من العبارات *(العطف ضمني بين العبارات)*:

$$\{(P\vee\neg R),(\neg Q\vee\neg R\vee P)\}$$

1 - 11 - الحل بالنقض Resfutation Resolution

يتمّ الحل بالنقض لإثبات صحة صيغة *wff* من مجموعة من الصيغ Δ باتباع الخطوات التالية:

1- تحويل كل صيغة موجودة في Δ إلى عطف عبارات (الشكل النظامي).

2- تحويل نفي الصيغة w والمطلوب إثباتها إلى عطف عبارات.

3- تجميع العبارات الناتجة عن الخطوتين السابقتين في مجموعة واحدة G.

4- نعاود تطبيق الحل على عبارات المجموعة Γ ونضيف ناتج الحل في كل مرة إلى المجموعة حتى الوصول إلى حالة لا يبقى فيها عبارات قابلة للحل أو الوصول إلى المجموعة الخالية.

مثال: عالم رفع الكتل

لتكن مجموعة العبارات Δ:

1. *BAT_OK*
2. $\neg$*MOVES*
3. *BAT_OK* $\wedge$ *LIFTABLE* $\Rightarrow$ *MOVES*

يُعطي تحويل العبارة الثالثة إلى الشكل النظامي:

4. $\neg$*BAT_OK* $\vee$ $\neg$*LIFTABLE* $\vee$ *MOVES*

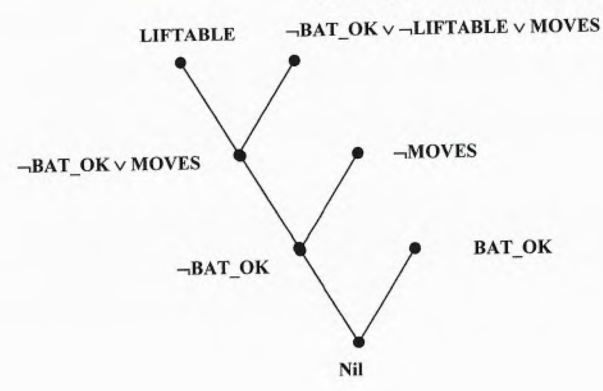
يُعطي نفي الصيغة المطلوب برهانها:

5. *LIFTABLE*

لنطبق الآن الحل للحصول على العبارات التالية:

6. $\neg$*BAT_OK* $\vee$ *MOVES (4,5 بحل)*
7. $\neg$*BAT_OK (6,2 بحل)*
8. *Nil (7,1 بحل)*

يُمكن تمثيل الحل بالشجرة التالية:



1 - 12 - أسئلة متعددة الخيارات

1. لتكن لدينا مجموعة القواعد التالية:

$P \Rightarrow (R \vee S)$

$\neg P \Rightarrow (R \vee S)$

$\neg S$

$(R \vee U) \Rightarrow Q$

يُمكن مما سبق برهان:

- ***a.*** ***<u>Q</u>***
- *b.* $\neg Q$
- *c.* S
- *d.* لا يوجد خيار صحيح

2. حدّد الصيغة غير القابلة للتحقيق *Unsatisfiable* فيما يلي:

- *a.* $((P \wedge Q) \Rightarrow (P \Rightarrow Q))$
- *b.* $((P \Rightarrow Q) \vee (P \wedge \neg Q))$
- *c.* $\neg(\neg (P \wedge Q) \Rightarrow (R \Rightarrow (\neg R \Rightarrow Q)))$
- ***d.*** **<u>لا يوجد خيار صحيح</u>**

3. لتكن لدينا مجموعة الحقائق التالية:

1. *"chose one of three roads: short, medium or long"*
2. *"the short road is always crowded"*
3. *"the medium road is not comfortable, but fast"*
4. *"the long road is comfortable"*
5. *"the chosen road should be comfortable."*

والتي يُمكن كتابتها كما يلي:

1. $short \lor medium \lor long$
2. $short \Rightarrow crowded$
3. $medium \Rightarrow (\neg comfortable \land fast)$
4. $long \Rightarrow comfortable$
5. $comfortable.$

يُمكن باستخدام الحقائق الخمسة السابقة برهان أن الطريق المختار هو الطريق الطويل *long*

a. ***false***

b. *true*

c. حسب طول الطريق

d. حسب سرعة السيارة

4. الشكل النظامي للصيغة التالية:

$$\neg\{[Smoke \Rightarrow Fire] \Rightarrow [(Smoke \wedge Heat) \Rightarrow Fire]\}$$

a. *true*

b. *false*

c. *Smoke Ú Fire*

d. غير ذلك

5. الشكل النظامي للصيغة التالية:

$$(\neg q \wedge (p \Rightarrow q)) \Leftrightarrow \neg p$$

a. <u>True</u>

b. *False*

c. *p*

d. <u>غير ذلك</u>

6. الشكل النظامي للصيغة التالية:

$$[(\neg P \vee \neg Q) \wedge (P \vee \neg Q) \wedge R] \Rightarrow [\neg Q \wedge R]$$

a. <u>True</u>

b. *False*

c. *R*

d. $P \vee R$

7. ليكن لدينا الصيغ الثلاث التالية:

$B \Rightarrow (C \wedge \neg D)$

$(C \wedge A) \Rightarrow \neg E$

$F \Rightarrow (D \vee E)$

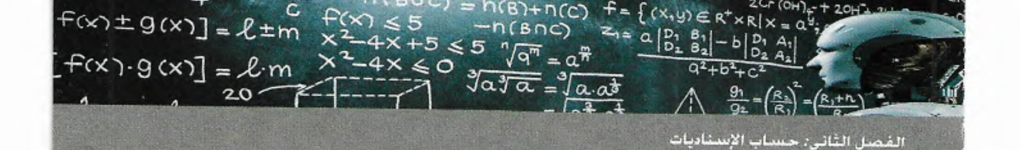
ما هي الصيغة التي يُمكن برهانها من الصيغ الثلاث السابقة:

a. <u>$A \Rightarrow (B \Rightarrow \neg F)$</u>

b. $(B \Rightarrow \neg F) \Rightarrow A$

c. $\neg A \Rightarrow (B \Rightarrow \neg F)$

d. ولا خيار



الفصل الثاني

حساب الإسناديات

The Predicate Calculus

2 - 1 - أهمية حساب الإسناديات

- لا يوفر حساب الفرضيات لغة غنية لتوصيف العالم.
- مثلاً: لا يمكن التعبير في حساب الفرضيات عن أن كل طلاب الصف العاشر نجحوا. أو أن بعض طلاب الصف العاشر نجحوا.

2 - 2 - الشكل *Syntax* - مكونات اللغة

- **مجموعة غير منتهية من الثوابت الغرضية *Object Constants***

Aa, 125, 13B, Q, John, EiffelTower

(تبدأ الثوابت الغرضية بحرف كبير).

- **مجموعة غير منتهية من الثوابت الوظيفية *Function Constants***

distanceBetween(x,y), times(x,y)

(تبدأ الثوابت الوظيفية بحرف صغير، كما نستخدم الأحرف الصغيرة للمتغيرات).

- **مجموعة من الثوابت العلاقاتية (الإسناديات) *Relation Constants***

Parent(x,y), Large(x), Clear(x)

- نستخدم أيضاً جميع الروابط $\vee,\wedge,\neg,\Rightarrow$ والمحددات *[,] ,(,)* والفاصلة ,.

الحدود *Terms*

- **كل ثابت غرضي هو حد.**
- **الثابت الوظيفي من الدرجة *n* متبوعاً بـ *n* حداً موضوعة بين قوسين ومفصولة بفواصل، هو أيضاً حد.**

أمثلة:

fatherOf)John)

times(4, plus(3,6))

الصيغ جيدة التركيب *wffs*

الذرات:

- تتألف من ثابت علاقاتي من البعد *n* متبوعاً بـ *n* حداً موضوعة بين قوسين ومفصولة بفواصل.
- يُمكن حذف الأقواس في حال كون بعد الثابت العلاقاتي *0*.
- **كل ذرة *Atom* هي صيغة جيدة التركيب *wff***

أمثلة:

Greatherthan(7,2), P(A,B,C,D), Q

الصيغ الفرضية جيدة التركيب *WFF*

يُدعى كل تعبير، يتشكل من صيغ حساب الإسناديات بالطريقة نفسها التي نُشكل بها صيغ حساب الفرضيات، بصيغة فرضية جيدة التركيب.

مثال:

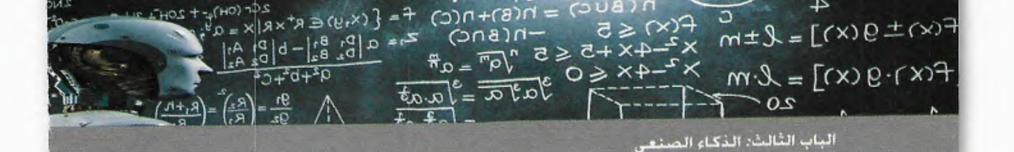
$[Greatherthan(7,2) \wedge Lessthan(15,4)] \vee \neg Brother(John,Sam) \vee P$

2 - 3 - الدلالة *Semantic*

- **يحقق تفسير صيغة *wff* إذا كان للصيغة القيمة *True* في هذا التفسير.**
- **نقول عن تفسير يحقق صيغة إنه نموذج لها.**
- **نقول عن صيغة تأخذ القيمة *True* من أجل كل التفاسير إنها صالحة *Valid*.**
- **نقول عن صيغة لا يوجد لها أي نموذج إنها غير قابلة للتحقيق *Unsatisfiable*.**
- **إذا كان لصيغة *w* القيمة *True* من أجل كل التفاسير التي تجعل لكل صيغة في مجموعة Δ القيمة *True* فنقول إن Δ تؤدي منطقياً لـ *w***

$$D|=w$$

- **تكون صيغتان متكافئتين إذ أخذتا قيم الحقيقة نفسها من أجل جميع التفسيرات. *(أي إذا وفقط إذا كانت كل منهما تؤدي للأخرى)*.**



2 - 4 - التكميم *Quantification*

لدينا المكمم العام $\forall$ *universal quantifier* والمكمم الوجودي $\exists$ *existential quantifier*

إذا كانت w صيغة و ξ متغيراً، فإن كلاً من:

$$(\forall \xi) w$$

$$(\exists \xi) w$$

هي صيغ *wff*

أمثلة

$$(\forall x)\ [P(x) \Rightarrow R(x)]$$

$$(\exists x)\ [P(x) \Rightarrow (\exists y)[R(x,y) \Rightarrow S(f(x))]]$$

بعض التكافؤات المفيدة

$$\neg (\forall \xi) w(\xi) \equiv (\exists \xi) \neg w(\xi)$$

$$\neg (\exists \xi) w(\xi) \equiv (\forall \xi) \neg w(\xi)$$

$$(\forall \xi) w(\xi) \equiv (\forall \eta) w(\eta)$$

الحل *Resolution*

إذا حوت عبارتان حرفيين متطابقين ومتتامين (أحدهما نفي الآخر). فيُمكن أن نحلهما بطريقة حساب الفرضيات نفسها.

إذا كان لدينا حرف λ (ξ) في عبارة (حيث ξ متغير) ، وحرفي متمم $\neg\lambda$ (τ) في عبارة أخرى (حيث τ هو حد لا يحوي المتغير ξ) . فيُمكن استبدال ξ بـ τ في العبارة الأولى، ومن ثم القيام بالحل للوصول لناتج حل *Resolvent* العبارتين.

مثال:

ليكن لدينا مثلاً العبارتان:

$$P(f(y),A) \vee Q(B,C)$$

و

$$\neg P(x,A) \vee R(x,C) \vee S(A,B)$$

نلاحظ أن العنصرين في أول كل عبارة متتامان وبالتالي يُمكن الحل على الحرفي *P(f(y),A)* لنحصل على ناتج الحل:

$$R(f(y),C) \vee S(A,B) \vee Q(B,C)$$

يتمّ حساب الاستبدالات المناسبة باستخدام إجرائية تُدعى التوحيد *Unification*.

لتوصيف هذه الإجرائية سنقوم أولاً بمناقشة مسألة الاستبدال *Substitution*.

2 - 5 - الاستبدال *Substitution*

نحصل على مُنتَسَخ الاستبدال *substitution instance* لتعبير باستبدال متغيرات التعبير بحدود.

مثال:

للتعبير *P[x,f(y),B]* منتسخات الاستبدال التالية:

- *P[z,f(w),B]*
- *P[x,f(A),B]*
- *P[g(z),f(A),B]*
- *P[C,f(A),B]*



يُدعى أول مُنتَسَخ بمغاير أبجدي *alphabetic variant* لأننا قمنا بشكل أساسي باستبدال متغيرات التعبير بمتغيرات أخرى.

يُدعى آخر مُنتَسَخ بالمنتسخ الأرضي *ground instance* إذ لا تحوي حدوده أي متغير.

يُمكن تمثيل أي استبدال بمجموعة مرتبة من الثنائيات:

$$s=\{\tau_1/\xi_1, \tau_2/\xi_2,..., \tau_n/\xi_n\}$$

حيث τ_i/ξ_i يعني أن كل ظهور للمتغير ξ_i سيُستبدل بالحد τ_i.

لاحظ أنه لا يُمكن استبدال متغير بحد يحوي هذا المتغير.

في المثال السابق قمنا باستخدام الاستبدالات التالية:

$s1=\{z/x, w/y\}$

$s2=\{A/y\}$

$s3=\{g(z)/x, A/y\}$

$s4=\{C/x, A/y\}$

لتدوين مُنتَسَخ استبدال لتعبير w باستخدام استبدال s نكتب ws.

وبهذا فإن:

$$P[z,f(w),B]=P[x,f(y),B]s1$$

عندما يُطبق الاستبدال s على كل عنصر في مجموعة من التعابير $\{wi\}$ ندون مجموعة منتسخات الاستبدال بـ

$$\{wi\}s$$



نقول إن مجموعة *{wi}* من التعابير قابلة للتوحيد *unifiable* إذا وجد استبدال *s* بحيث يكون:

w1s=w2s=w3s=...

نقول في هذه الحالة إن *s* هو الموحد *unifier* لـ *{wi}*

مثال:

s={A/x,B/y}

يوحد المجموعة: *{[P[x,f(y)),B], P[x,f(B),B}* ليُعطي: *{[P[A,f(B),B}*

2 - 6 - خوارزمية التوحيد Algorithm Unification

نستخدم الخوارزمية *UNIFY* والتي تعمل على تعابير موضوعة بشكل قائمة مهيكلة.

فمثلاً التعبير:*(¬P(x,f(A,y))* يُكتب بالشكل: *(¬P x (f A y))*

ويكون *¬P* التعبير الأول و *(f A x)* التعبير الثالث.

مجموعات عدم التوافق Disagreement Set

نستخدم في الخوارزمية *UNIFY* مفهوم مجموعات عدم التوافق *.disagreement set*

نحصل على مجموعة عدم التوافق لمجموعة غير خالية *W* من التعابير بتحديد موقع أول رمز (بدءاً من اليسار) لا تكون عنده رموز كل التعابير متساوية.

ومن ثم نستخرج من كل تعبير التعابير الجزئية التي تبدأ بالرمز الذي يحتل هذا الموقع.

تكوّن هذه التعابير الجزئية مجموعة عدم التوافق لـ W.

مثلاً إذا كانت:

$$W=\{(\neg P\ x\ (f\ A\ y)),(\neg P\ x\ (f\ z\ B)\}$$

فإن مجموعة عدم التوافق لها هي:

$$\{A,z\}$$

خوارزمية التوحيد

UNIFY(Γ)

(حيث Γ مجموعة تعابير لها بنية القائمة)

$1. k \leftarrow 0;\ G_k \leftarrow \Gamma; \alpha_k \leftarrow e$

(الخطوة الابتدائية: ε هو الاستبدال الفارغ)

2. If Singleton(G_k) Then Return(α_k)

(إذا كانت G_k أحادية نخرج من الإجراء و نُرجع α_k)

3. $D_k \leftarrow$ DisagreementSet(G_k)

(نحسب D_k مجموعة عدم التوافق لـ G_k)

4. If Exits(v_k,t_k) in D_k and

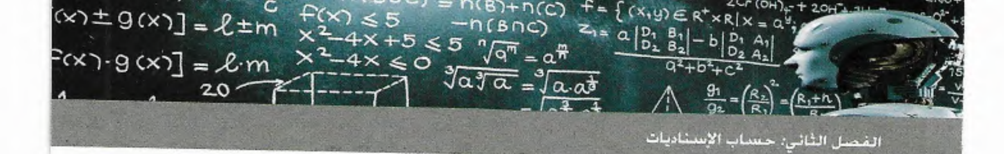
Variable(v_k) and Not Occur(v_k,t_k) Then

(إذا وجد v_k و t_k في المجموعة D_k حيث v_k متغير لا يظهر في t_k)

Continue (نستمر)

Else (و إلا فإن Γ غير قابلة للتوحيد. ونخرج)

Return (FAILURE)



أمثلة:

منتسخ الاستبدال المشترك الأعم	مجموعات من التعابير
$P(A)$	$\{P(x),P(A)\}$
$P[f(x),x,g(x)]$	$\{P[f(x),y,g(y)],P[f(x),z,g(x)]\}$
$P[f(x,g(A,y)),g(A,y)]$	$\{P[f(x,g(A,y)),g(A,y)],P[f(x,z),z]\}$

الحل *Resolution*

لنفرض أن لدينا العبارتين γ_1 و γ_2 (كل عبارة *Clause* ممثلة بمجموعة من الحرفيات *Literals*)

إذا وجد ذرة *atom* Φ في γ_1 وحرفي *literal* $\neg\Psi$ في γ_2 بحيث أن لـ Φ و Ψ الموحد الأعم μ . فإن لهاتين العبارتين ناتج حل *resolvent* ρ نحصل عليه بتطبيق الاستبدال على اجتماع γ_1 و γ_2 بعد حذف العنصرين المتتامين Φ و $\neg\Psi$

أي:

$$\rho = [(\gamma_1 - \{\Phi\}) \cup (\gamma_2 - \{\neg\Psi\})]\mu$$

نقوم قبل الحل بإعادة تسمية المتغيرات في العبارات بحيث لا يظهر أي متغير لعبارة في عبارة أخرى.

مثال:

إذا كان المطلوب حل:

$P(x) \vee Q(f(x))$

مع:

$R(g(x)) \vee \neg Q(f(A))$

فإننا نقوم أولاً بإعادة كتابة العبارة الثانية لنحصل على:

$R(g(y)) \vee \neg Q(f(A))$

تُدعى عملية إعادة تسمية المتغيرات بتقييس المتغيرات *Standardizing*

أمثلـــة

1- ينتج عن حل: $\{(P(x),Q(x,y)\}$ و $\{(\neg P(A),R(B,z)\}$

المجموعة:

$\{Q(A,y),R(B,z)\}$

2- يُمكن حل:

$\{P(x,x),Q(x),R(x)\}$ و $\{\neg P(A,z), \neg Q(B)\}$

بطريقتين مختلفتين لنحصل على:

$\{Q(A),R(A), \neg Q(B)\}$

و

$\{P(B,B),R(B), \neg P(A,z)\}$

2 - 7 - تحويل الصيغ إلى عبارات *Clause Form*

خطوات تحويل الصيغ إلى عبارات

يُمكن وضع أي صيغة *wff* بشكل عبارة *Clause Form* باتباع الخطوات التالية:

(1) حذف إشارات الاقتضاء (مثل حساب الفرضيات).

(2) تقليص مجالات النفي (مثل حساب الفرضيات).

(3) تقييس المتغيرات.

بما أن المتغيرات الواقعة ضمن مجال المكممات هي متغيرات شكلية *Dummy*، فيُمكن إعادة تسميتها بحيث أن لكل مكمم متغيراته الخاصة به.

مثلاً يُمكن كتابة الصيغة:

$$(\forall x)[\neg P(x) \vee (\exists x) Q(x)]$$

كـ

$$(\forall x)[\neg P(x) \vee (\exists y) Q(y)]$$

(4) حذف المكممات الوجودية

لاحظ أنه في:

$$(\forall x)[(\exists y) Height(x,y)]$$

يتواجد المكمم الوجودي ضمن مجال المكمم العام، وبهذا فإن y يُمكن أن يتعلق بقيمة x

مثلاً إذا كان المعنى الموافق للتعبير السابق هو:

«لكل شخص x الطول y"

فمن الواضح أن الطول يتعلق بالشخص. يُمكن إظهار هذا الارتباط بتعريف دالة غير معروفة *h(x)*، والتي تقوم بمقابلة كل قيمة لـ *x* بقيمة *y* الموجودة تُدعى مثل هذه الدالة بدالة سكوليم ***Skolem function*** نسبة إلى عالم المنطق *Thoralf Skolem*.

إذا استعملنا دالة سكوليم عوضاً عن المتغير *y* فيُمكن أن نحذف المكمم الوجودي لنحصل على:

$$(\forall x)[\,Height(x,h(x))]$$

وتكون القاعدة العامة لحذف المكممات الوجودية أن نستبدل كل متغير مكمم وجودياً بدالة سكوليم.

هذه الدالة هي المتغيرات المكممة*arguments* تكون محدّدات باستخدام $\forall$ والتي يحوي مجالها المتغير المراد حذفه.

يجب أن تكون أسماء دوال سكوليم المستخدمة جديدة وغير مستخدمة في الصيغ.

مثال: يُمكن حذف $(\exists z)$ من:

$$[(\forall w Q(w)] \Rightarrow (\forall x)\{(\forall y)\{(\exists z)[P(x,y,z) \Rightarrow (\forall u)R(x,y,u,z)]\}\}$$

للحصول على:

$$[(\forall w Q(w)] \Rightarrow (\forall x)\{(\forall y)\,[P(x,y,g(x,y)) \Rightarrow (\forall u)R(x,y,u,g(x,y))]\}$$

مثال: يُمكن حذف $(\exists w)$ من:

$$(\forall x)\{\neg P(x) \vee \{(\forall y)[\neg P(y) \vee P(f(x,y))] \wedge (\exists w)[Q(x,w) \wedge \neg P(w)]\}\}$$

لنحصل على:

$$(\forall x)\{\neg P(x) \vee \{(\forall y)[\neg P(y) \vee P(f(x,y))] \wedge [Q(x,h(x)) \wedge \neg P(h(x)]\}\}$$

حيث $g(x,y)$ و $h(x)$ هي دوال سكوليم

إذا لم يكن المتغير المكمم وجودياً ضمن مجال أي مكمم عام، فإننا نستخدم دالة سكوليم بدون محددات أي ثابت

وبهذا فإن: $(\exists x)P(x)$ تصبح: $P(SK)$

حيث يرمز الثابت SK لمقدار نعرف أنه موجود ويجب أن يكون ثابتاً جديداً لا يُستخدم في الصيغ.

لحذف المتغيرات المكممة وجودياً في مجموعة من الصيغ نُطبّق الإجراء السابق على كل صيغة.

ندعو في النهاية مجموعة الصيغ الناتجة بعد ذلك بشكل سكوليم *Skolem form*.

(5) التحويل إلى الشكل الأمامي *Prenex Form*

نضع جميع المكممات العامة في بداية الصيغة ونجعل مجال كل مكمم يضم كامل الصيغة.

مثال:

الشكل الأمامي للصيغة السابقة:

$$(\forall x)\{\neg P(x)\vee\{(\forall y)[\neg P(y)\vee P(f(x,y))]\wedge[Q(x,h(x))\wedge\neg P(h(x)]\}\}$$

يكون:

$$(\forall x)(\forall y)\{\neg P(x)\vee\{[\neg P(y)\vee P(f(x,y))]\wedge[Q(x,h(x))\wedge\neg P(h(x)]\}\}$$

(6) وضع المصفوفة في شكل العطف النظامي *Conjunctive Normal Form*

استخدم الطريقة نفسها المتبعة في حساب الفرضيات لوضع المصفوفة بشكل العطف النظامي وذلك بتكرار تطبيق أحد قوانين التوزيع.

أي عملياً استبدال كل:

$(w1 \vee w2) \wedge (w1 \vee w3)$ بـ $w1 \vee (w2 \wedge w3)$

مثال: تُصبح الصيغة السابقة:

$$(\forall x)\ (\forall y)\{\neg P(x) \vee \{[\neg P(y) \vee P(f(x,y))] \wedge [Q(x,h(x)) \wedge \neg P(h(x)]\}\}$$

بعد وضعها في شكل العطف النظامي:

$$(\forall x)\ (\forall y)\{[\neg P(x) \vee \neg P(y) \vee P(f(x,y))] \wedge$$

$$[\neg P(x) \vee Q(x,h(x))] \wedge$$

$$[\neg P(x) \vee \neg P(h(x))]\}$$

(7) حذف المكممات العامة

يُمكن الآن حذف كل المكممات العامة واصطلاح أن كل متغيرات المصفوفة مكممة تكميماً عاماً. وبهذا يتبقى لدينا الآن مصفوفة في شكل العطف النظامي.

(8) حذف إشارات العطف $\wedge$

يُمكن الآن حذف $\wedge$ وذلك باستبدال التعابير من الشكل $(w1 \wedge w2)$ بمجموعة الصيغ جيدة التركيب $\{w1,w2\}$.

نحصل إذاً بعد عدة استبدالات على مجموعة منتهية من الصيغ والتي كل منها فصل بين حرفيات. ندعو هذا الشكل بالعبارة *Clause*.

يُمكن تحويل المثال السابق إلى مجموعة العبارات:

$$\neg P(x) \vee \neg P(y) \vee P[f(x,y)]$$

$$\neg P(x) \vee Q[x,h(x)]$$

$$\neg P(x) \vee \neg P[h(x)]$$

(9) إعادة تسمية المتغيرات

يُمكن إعادة تسمية المتغيرات بحيث لا يظهر أي متغير في أكثر من عبارة.

$\neg P(x1) \vee \neg P(y) \vee P[f(x1,y)]$

$\neg P(x2) \vee Q[x2,h(x2)]$

$\neg P(x3) \vee \neg P[h(x3)]$

مثال: مسألة الربوط موزع الطرود

لنفرض أن هذا الربوط يعرف أن جميع الطرود الموجودة في الغرفة *27* أصغر من أي طرد موجود في الغرفة *28*. أي:

1. $(\forall x,y)\{[Package(x) \wedge Package(y) \wedge Inroom(x,27) \wedge Inroom(y,28)] \supset Smaller(x,y)\}$

لنختصر أسماء الثوابت لنجعل الصيغة أكثر صغراً ولنحولها إلى شكل العبارة ما يُعطي:

2. $\neg P(x) \vee \neg P(y) \vee \neg I(x,27) \vee \neg I(y,28) \vee S(x,y)$

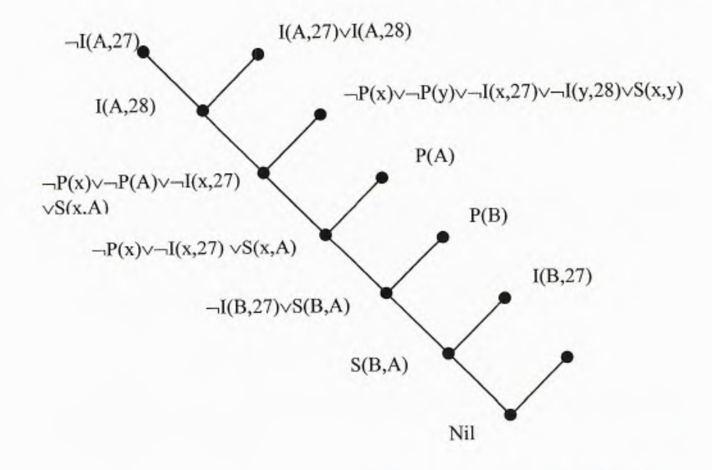
لنفرض أن الربوط يعرف أن الطرد *A* موجود إما في الغرفة *27* وإما في الغرفة *28* إلا أنه لا يعرف في أي منهما.

كما أن الربوط يعرف أن الطرد *B* موجود في الغرفة *27* كما أنه ليس أصغر من الطرد *A*.

3. $P(A)$
4. $P(B)$
5. $I(A,27) \vee I(A,28)$
6. $I(B,27)$
7. $\neg S(B,A)$

يُمكن للربوط، باستخدام الحل بالنقض، إثبات أن الطرد *A* موجود في الغرفة *27*.

يُبين الشكل التالي شجرة البرهان.



إيجاد الحل *Answer Extraction*

مثال: مسألة الربوط موزع الطرود

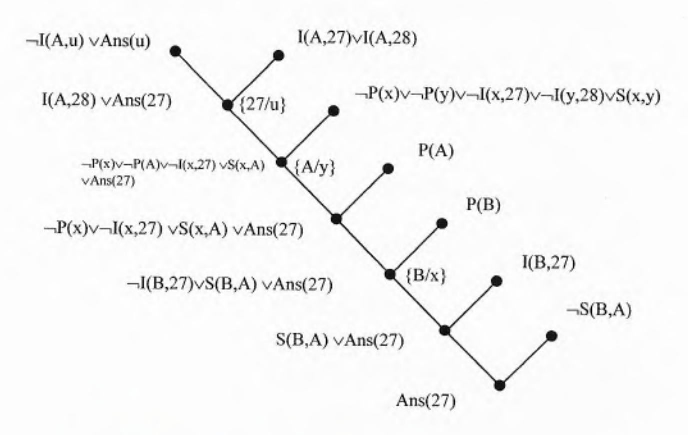
«في أي غرفة توجد الكتلة *A*"

تُبين الشجرة التالية كيفية استخدام الحرفي *Ans* لبرهان الصيغة:

$$(\exists u)I(A,u)$$

والتي يُمكن عدّها التعبير عن سؤال الربوط لنفسه:

«في أي غرفة توجد الكتلة *A*".



2 - 8 - تمثيل المعارف باستخدام المنطق من الدرجة الأولى

أمثلة

- *All crows are black*

$$\forall x\ Crow(x) \Rightarrow Black(x)$$

- *Mary likes the color of one of John's ties*

$$\exists x\ Like(Mary, color(x)) \wedge Tie(x) \wedge Owner(x,John)$$

- *Every gardener likes the sun*

$$(\forall x)\ gardener(x) \Rightarrow likes(x,Sun)$$

- *You can fool some of the people all of the time*

$$(\exists x)(\forall t)\ (person(x) \wedge time(t)) \Rightarrow can\text{-}fool(x,t)$$

- *You can fool all of the people some of the time*

$$(\forall x)(\exists t)\ (person(x) \wedge time(t) \Rightarrow can\text{-}fool(x,t)$$

- *All purple mushrooms are poisonous*

$$(\forall x)\ (mushroom(x) \wedge purple(x)) \Rightarrow poisonous(x)$$

- *No purple mushroom is poisonous*

$$\neg(\exists x)\ purple(x) \wedge mushroom(x) \wedge poisonous(x)$$

or, equivalently,

$$(\forall x)\ (mushroom(x) \wedge purple(x)) \Rightarrow \sim poisonous(x)$$

- *There are exactly two purple mushrooms*

$$(\exists x)(\exists y)\ mushroom(x) \wedge purple(x) \wedge mushroom(y) \wedge purple(y) \wedge \neg(x=y) \wedge (\forall z)\ (mushroom(z) \wedge purple(z)) \Rightarrow ((x=z)\ v\ (y=z))$$

- *X is above Y if X is on directly on top of Y or else there is a pile of one or more other objects directly on top of one another starting with X and ending with Y*

$$(\forall x)(\forall y)\ above(x,y)\ \hat{U}\ (on(x,y)\ v\ (\exists z)\ (on(x,z) \wedge above(z,y)))$$

2 - 9 - أسئلة متعددة الخيارات

1 - ليكن لدينا:

(1) $P(x,y,y)$, $P(f(C),C,v)$

(2) $P(x, f(x))$, $P(x,y)$, $P(g(u),u)$

(3) $p(a,X)$, $p(Y, f(Y))$

(4) $parents(x,father(x),mother(Bill)),parents(Bill,father(y),z)$

حدّد العبارة الخاطئة فيما يلي:

a. الفقرتين في *(1)* قابلة للتوحيد

***b*. الفقرات الثلاث في *(2)* قابلة للتوحيد**

c. الفقرتين في *(3)* قابلة للتوحيد

d. الفقرتين في *(4)* قابلة للتوحيد

2 - لتكن لدينا مجموعة القواعد التالية:

$\forall\ x, y, z\ S(x,z) \wedge S(y,z) \Rightarrow U(x,y) \wedge U(y,x)$

$\forall x, y\ U(x, y) \Rightarrow F(x, y)$

$\forall x, y, z\ F(x, y) \wedge F(y, z) \Rightarrow F(x, z)$

ولتكن لدينا الحقائق التالية:

$S(A, F)$

$S(B, F)$

$S(B, G)$

$S(C, G)$

ولتكن لدينا الأهداف الثلاثة التالية:

$G1: F(A, B)$

$G2: F(A, C)$

$G3: F(B, C)$

حدّد فيما يلي الصيغة التي لا يُمكن برهانها من مجموعة القواعد والحقائق السابقة:

a. $G1 \land G2 \land G3$

b. <u>$G1 \land \neg G2 \land G3$</u>

c. $G1 \lor \neg G2 \lor G3$

d. $G1 \lor G2 \lor G3$

3 - لتكن لدينا الصيغة التالية:

$$(\forall x \exists y\, P(x) \Rightarrow (Q(x) \land M(y, x))) \lor \neg (\exists x\, R(x) \land S(x))$$

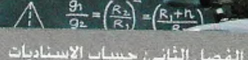
عند وضع الصيغة بالشكل النظامي ينتج ثوابت سكوليم من الشكل:

a. <u>$f(x)$</u>

b. C

c. $f(x,y)$

d. ولاخيار مما سبق



4 - لتكن لدينا مجموعة القواعد التالية:

$$\forall x\ \forall y\ \forall z\ (R(x,y) \wedge R(x, z)) \Rightarrow R(y,z)$$

$$\forall x\ \forall y\ R(x,y) \Rightarrow \exists u\ (R\ (u,x) \wedge R(u,y)\ )$$

يُمكن مما سبق برهان:

$$\forall x\ \forall y\ R(x,y) \Rightarrow R(y,x)$$

a. **صح**

b. خطأ

c. بعض الأحيان

d. حسب قيمة ثوابت سكوليم

5 - حدّد فيما يلي الصيغة الصالحة *valid*

a. $(\forall x, y\ (\ p(x, y) \Rightarrow p(y, x))\ ) \Rightarrow \forall z\ p(z, z)$

b. $\forall x, y\ (\ p(x, y) \Rightarrow (p(y, x) \Rightarrow \forall z\ p(z, z)\ )\ )$

c. $(\forall x\ p(x)\ ) \Rightarrow \exists y\ p(y)$

d. **ولا خيار**

6 - ليكن لدينا الصيغ الثلاث التالية:

$P(x) \vee Q(F(x), x)$

$R(y) \vee Q(y, z)$

$R(F(A))$

ما هي الصيغة التي لا يُمكن برهانها من الصيغ الثلاث السابقة:

a. $P(A)$

b. $\exists x\ P(x)$

c. $\forall\ x\ P(x)$

d. **ولا خيار مما سبق**

7 - ليكن لدينا القواعد التالية:

$Guilty(b) \Rightarrow \neg Guilty(a)$

$Guilty(b) \Rightarrow \neg Guilty(c)$

$Guilty(a) \Rightarrow \neg Guilty(c)$

$Knows(a, d) \Rightarrow Lies(b)$

$\neg Knows(a, d) \Rightarrow Lies(a)$

$Knows(a, d) \Rightarrow Lies(c)$

$Lies(x) \Rightarrow Guilty(x)$

يُمكن من هذه القواعد برهان:

- **a.** ***Guilty(a)***
- *b.* *Guilty(b)*
- *c.* *Guilty(c)*
- *d.* *Guilty(d)*

8 - أي مما يلي هو نمذجة للحقيقة التالية:

John has exactly one brother

- *a.* $\exists x, y\ brother(John, x) \wedge brother(John, y) \wedge x = y$
- *b.* $\exists x\ brother(John, x) \Rightarrow \forall y(brother(John, y) \Rightarrow x = y)$
- *c.* $\exists x\ brother(John, x) \Rightarrow \forall y(brother(John, y) \wedge x = y)$
- **d.** **غير ذلك**

9 - لتكن لدينا مجموعة القواعد التالية:

$\forall$ ***x, y, z SpeakLang(x,z)***$\wedge$***SpeakLang(y,z)*** $\Rightarrow$***Understand(x,y)*** $\wedge$***Understand(y,x)***

$\forall$***x, y Understand(x, y)*** $\Rightarrow$ ***Friend(x, y)***

$\forall$***x, y, z Friend(x, y)*** $\wedge$ ***Friend(y, z)*** $\Rightarrow$ ***Friend(x, z)***

ولتكن لدينا الحقائق التالية:

SpeakLang(Ann, French)

SpeakLang(Bob, French)

SpeakLang(Bob, German)

SpeakLang(Cal, German)

ولتكن لدينا الأهداف الثلاث التالية:

G1: Friend(Ann, Bob)

G2: Friend(Ann, Cal)

G3: Friend(Bob, Cal)

حدّد فيما يلي الصيغة التي لا يُمكن برهانها من مجموعة القواعد والحقائق السابقة:

a. $G1 \wedge G2 \wedge G3$

b. $\underline{\mathbf{G1 \wedge \neg G2 \wedge G3}}$

c. $G1 \vee \neg G2 \vee G3$

d. $G1 \vee G2 \vee G3$

10 - لتكن لدينا مجموعة القواعد التالية:

$$\forall x\ \forall y\ (\ p(x,y)\ \Rightarrow \exists z\ q(x,y,z)\ )$$

$$\exists x\ \forall y\ \forall z\ (r(y,z)\ \Leftrightarrow q(x,y,z)\ )$$

$$\forall x\ \exists y\ (\neg p(x,y) \Rightarrow \forall z\ q(x,y,z))$$

عند وضع القواعد السابقة بالشكل النظامي ينتج ثوابت سكوليم التالية:

a. <u>*C, g(x), f(x, y)*</u>

b. *C1, C2, g(x)*

c. *C1, C2, f(x,y)*

d. ولاخيار مما سبق

11 - لتكن لدينا مجموعة القواعد التالية:

$$\forall x\ \forall y\ (\ p(x,y)\ \Rightarrow \exists z\ q(x,y,z)\ )$$

$$\exists x\ \forall y\ \forall z\ (r(y,z)\ \Leftrightarrow q(x,y,z)\ )$$

$$\forall x\ \exists y\ (\neg p(x,y) \Rightarrow \forall z\ q(x,y,z))$$

يُمكن مما سبق برهان:

$$\exists w\ \exists x\ \exists y\ \exists z\ )\ r(x,y) \wedge q(x,w,z)\ )$$

a. <u>صح</u>

b. خطأ

c. بعض الأحيان

d. حسب قيمة ثوابت سكوليم

12 - حدّد عدد الحالات غير قابلة للتوحيد من الثنائيات الأربعة التالية:

p(g(y), x, f(g(y)) and p(z, h(z,w), f(w))

R(f(x), y) and R(z, g(w))

R(f(x), x) and R(y, g(y))

P[f(x),y,g(y)] and P[f(x),z,g(x)]

a. *0*

b. ***1***

c. *2*

d. *3*

13 - ليكن لدينا المعارف التالية:

∀*x Person(x) ^Wood(x) Þ Witch(x)*

∀*y Duck(y) Þ Wood(y)*

∀*x, y Duck(y) ^ Equals(weight(x),weight(y)) Þ Wood(x)*

Equals(weight(A),weight(D))

Person(A)

Duck(D)

يُمكن من هذه المعارف برهان:

a. ***Witch(A)***

b. *¬Witch(A)*

c. *Witch(D)*

d. ولا خيار



الفصل الثالث

النظم الخبيَرة

Expert Systems

- تُعتبر النظم الخبيرة من أكثر تطبيقات تقانات المحاكمة في الذكاء الصنعي، والتي تستخدم الحقائق والقواعد لتضمين المعارف حول حقل معين من حقول المعارف البشرية مثل الطب والهندسة والأعمال.
- تُصمم النظم الخبيرة عموماً لحل مسائل التصنيف واتخاذ القرارات مثل (التشخيص الطبي، الوصفات العلاجية، تنظيم البورصات، وغيرها ...).
- النظم الخبيرة هي أدوات ذكاء صنعي، وهذا يعني أننا لا نستعملها إلا في المسائل التي ليس لها أي خوارزمية واضحة أكيدة لحلها.
- تتطلب النظم الخبيرة، وجود خبرة نودُّ نمذجتها. أي أنه، لا معنى للنظم الخبيرة إلا في المجالات التي توجد فيها خبرةٌ بشرية. والخبير هو الشخص الذي يعرف مجال التطبيق، ويعرف، نوعاً ما، كيف ينقل معرفته للآخرين.

التفكير البشري معقد جداً ومن الصعب تمثيله بخوارزمية. ومع ذلك فإن معظم الخبراء قادرين على وضع معارفهم لحل المسائل على شكل قواعد من الشكل (إلا أنهم غير قادرين على ضمان كمال هذه القواعد ولا تكاملها معاً!):

IF <antecedent> مقدمة *THEN <consequent>* النتيجة



مثال تعليمي:

يُمكن أن نتصور مثلاً موظف في بنك مسؤول عن إعطاء القروض للمتعاملين. يستخدم نظام خبير يُساعده في تقرير فيما إذا كان من المناسب إعطاء قرض لمتعامل. لنفرض أن الذرات التالية تستخدم المعاني الموافقة:

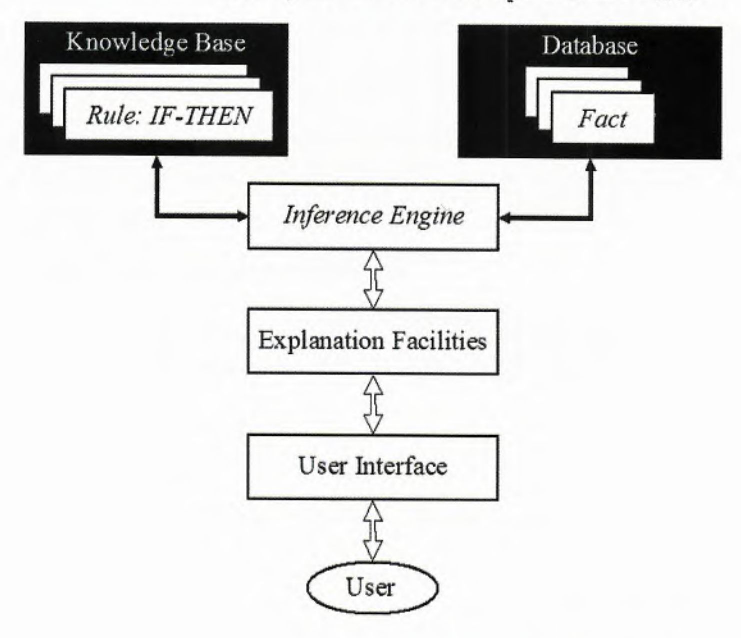
OK	يجب أن يتمّ الموافقة على القرض
COLLAT	ضمانة القرض مقنعة
PYMT	العميل قادر على تسديد دفعات القرض
REP	للعميل سمعة مالية جيدة
APP	تخمين الضمانة أكبر من مبلغ القرض
RATING	للعميل دفعات دورية منتظمة
INC	يتجاوز دخل العميل مصاريفه
BAL	للعميل صفحة متوازنة ممتازة

يُمكن استخدام القواعد التالية بهدف إقرار القرض:

1. $COLLAT \wedge PYMT \wedge REP \Rightarrow OK$
2. $APP \Rightarrow COLLAT$
3. $RATING \Rightarrow REP$
4. $INC \Rightarrow PYMT$
5. $BAL \wedge REP \Rightarrow OK$

3 - 1 - البنية العامة للنظام الخبير

يُبين الشكل التالي البنية العامة للنظام الخبير:



- **قاعدة المعرفة *Knowledge Base***

تحوي قاعدة المعرفة معارف المجال والمفيدة في حل المسائل. تُمثّل المعارف على شكل قواعد استنتاج.

IF <*antecedent*> مقدمة *THEN* <*consequent*> النتيجة

عندما يكون الشرط محققاً يتمّ تطبيق القاعدة وتنفيذ الفعل الموافق.

- **قاعدة البيانات** ***Data Base***

تحوي قاعدة البيانات مجموعة من الحقائق *Facts* والتي تُستخدم كدخل لشروط القواعد المخزنة في قاعدة المعرفة.

- **محرك الاستدلال** ***Inference engine***

يقوم محرك الاستدلال بالمحاكمة اللازمة للوصول إلى الحل، عبر تطبيق قواعد قاعدة المعرفة على حقائق قاعدة البيانات لاستنتاج حقائق جديدة.

- **نظام الشرح** ***Explanation System***

يتنصت نظام الشرح على محرك الاستدلال ليعطي للمستخدم تفسيراً لكيفية وصوله للحقائق الجديدة انطلاقاً من حقائق قاعدة البيانات.

- **واجهة الاستخدام** ***User Interface***

وهو وسيلة التخاطب بين المستخدم والنظام الخبير.

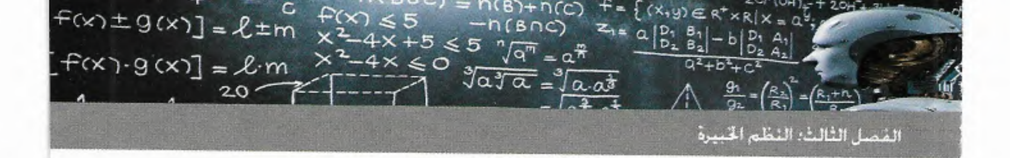
3 - 2 - استراتيجيات محرك الاستدلال

تُمثّل المعرفة في النظم الخبيرة باستخدام القواعد *Rules*، وتوضع البيانات على شكل حقائق *Facts*.

يقوم محرك الاستدلال بمقارنة كل قاعدة موجودة في قاعدة المعرفة مع الحقائق الموجودة في قاعدة البيانات، وعندما يجد المحرك أن شرط قاعدة محقق يقوم بتنفيذ القاعدة.

تولد عمليات المطابقة والتنفيذ سلسلة معينة ندعوها سلسلة الاستدلال *chains Inference*.

تُحدّد سلسلة الاستدلال كيفية تطبيق القواعد للوصول إلى نتيجة.



3 - 3 - السَلسَلة الأمامية *Forward Chaining*

تعتمد هذه الاستراتيجية على الفكرة التالية: «ولّد ما يُمكن توليده حتى الوصول للنتيجة المطلوبة».

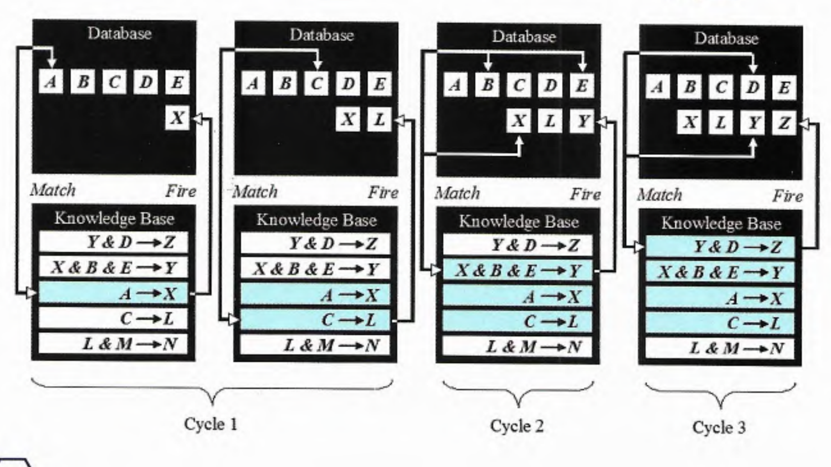
يقوم محرك الاستدلال في هذه الاستراتيجية وفي كل دورة *Cycle* بمسح القواعد لتحديد القواعد القابلة للتطبيق مع قاعدة الحقائق.

يُطبّق محرك الاستدلال هذه القواعد اعتباراً من القاعدة الأعلى مما سيُضيف حقائق جديدة إلى قاعدة الحقائق.

ثم ينتقل إلى الدورة التالية، وهكذا، حتى توليد النتيجة المطلوبة.

- تولّد السَلسَلة الأمامية كل ما يُمكن توليده.
- يُمكن في السَلسَلة الأمامية تطبيق قواعد لا علاقة لها بالمسألة المطلوبة.
- يُمكن في بعض الأحيان وحين يكون الهدف استنتاج حقيقة معينة ألا تكون هذه الاستراتيجية فعالة.

مثال *1*:



مثال 2:

ليكن لدينا قاعدة الحقائق $\{B,C\}$، والقواعد:

$(1)\ B \wedge D \wedge E \Rightarrow F$

$(2)\ G \wedge D \Rightarrow A$

$(3)\ C \wedge F \Rightarrow A$

$(4)\ B \Rightarrow X$

$(5)\ D \Rightarrow E$

$(6)\ X \wedge A \Rightarrow H$

$(7)\ C \Rightarrow D$

$(8)\ X \wedge C \Rightarrow A$

$(9)\ X \wedge B \Rightarrow D$

والمطلوب استنتاج الحقيقة H.

في الدورة الأولى: القاعدة 4 والقاعدة 7 قابلتين للتطبيق.

تضيف القاعدة 4 الحقيقة X إلى قاعدة الحقائق، وتضيف القاعدة 7 الحقيقة D إلى قاعدة الحقائق.

في الدورة الثانية: القاعدة 8 والقاعدة 5 قابلتين للتطبيق.

تضيف القاعدة 8 الحقيقة A إلى قاعدة الحقائق، وتضيف القاعدة 5 الحقيقة E إلى قاعدة الحقائق.

في الدورة الثالثة: القاعدة 1 والقاعدة 7 قابلتين للتطبيق.

تضيف القاعدة 1 الحقيقة F إلى قاعدة الحقائق، وتضيف القاعدة 7 الحقيقة H إلى قاعدة الحقائق.

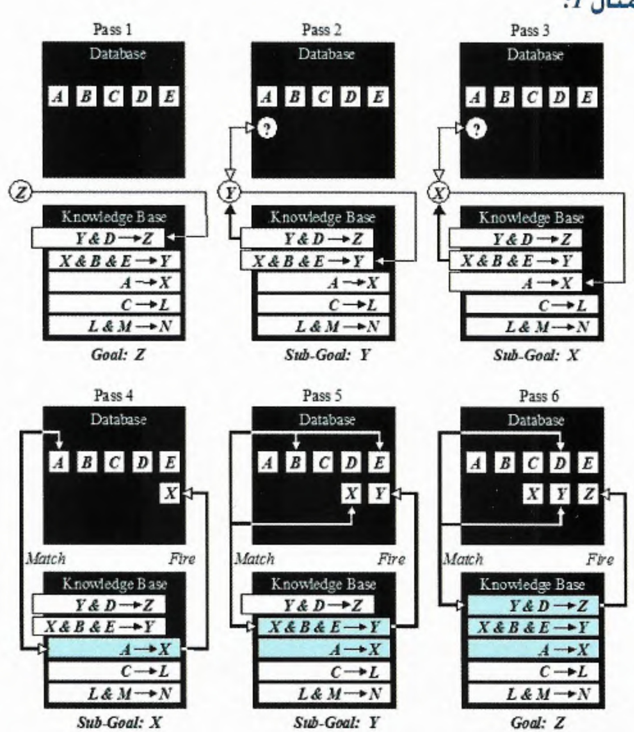
وبذلك تتحقق H وتتوقف عملية البحث.

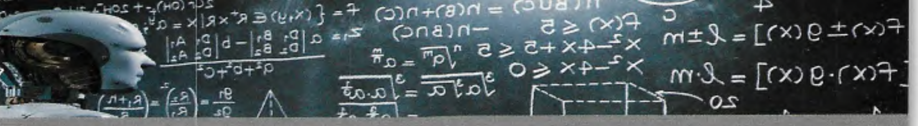
3 - 4 - السَلسَلة الخلفية Backward Chaining

تعتمد هذه الاستراتيجية على الفكرة التالية: "ابدأ من الهدف وحاول برهان كل ما يلزم لتحقيقه".

تعتمد آلية السَلسَلة الخلفية على البدء من الحقيقة الهدف، والبحث في مجموعة القواعد عن القواعد التي تقع هذه الحقيقة في نتيجتها، ومن ثم إنشاء قائمة بالحقائق الواجب برهانها لنتمكن من تطبيق القواعد السابقة ثم نعاود تطبيق هذه الآلية عودياً على الحقائق الموجودة في هذه القوائم. وهكذا.

مثال 1:





مثال 2:

ليكن لدينا قاعدة الحقائق $\{B,C\}$، والقواعد:

$$(1)\ B \wedge D \wedge E \Rightarrow F$$

$$(2)\ G \wedge D \Rightarrow A$$

$$(3)\ C \wedge F \Rightarrow A$$

$$(4)\ B \Rightarrow X$$

$$(5)\ D \Rightarrow E$$

$$(6)\ X \wedge A \Rightarrow H$$

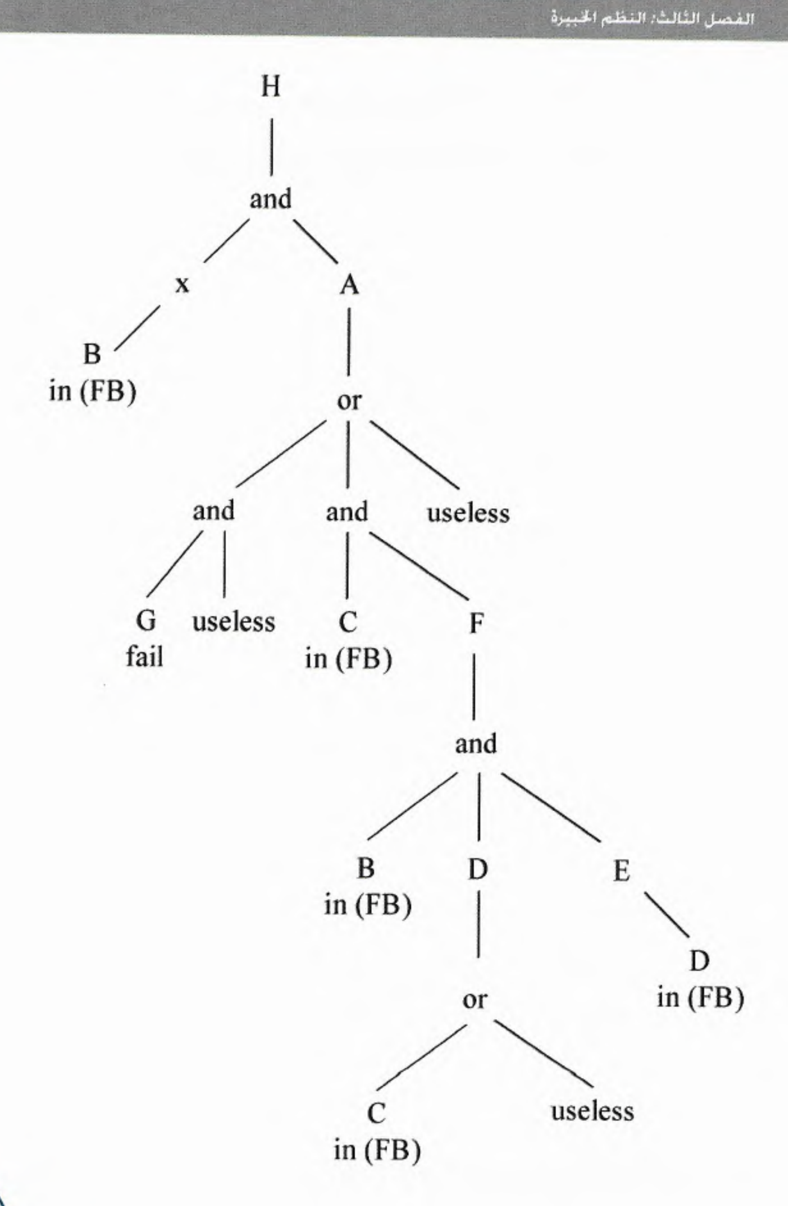
$$(7)\ C \Rightarrow D$$

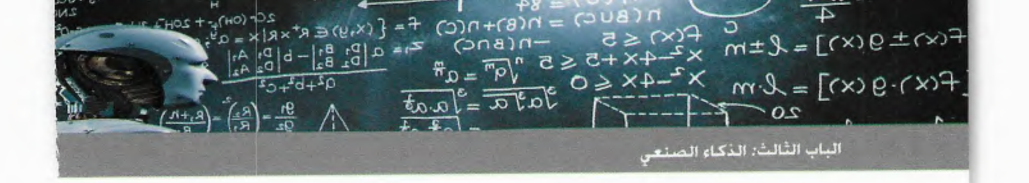
$$(8)\ X \wedge C \Rightarrow A$$

$$(9)\ X \wedge B \Rightarrow D$$

والمطلوب استنتاج الحقيقة H.

يمكن توصيف تنفيذ خوارزمية السَلسَلة الخلفية بشجرة عقدها إما حقائق وإما إحدى *and* أو *or*. نُسمّي هذه الشجرة، شجرة *and-or*.


H
and
x
A
B
in (FB)
or
and
and
useless
G
fail
useless
C
in (FB)
F
and
B
in (FB)
D
E
or
D
in (FB)
C
in (FB)
useless



3 - 5 - حل التضارب *Conflict Resolution*

ليكن لدينا القواعد التالية في قاعدة المعرفة:

Rule 1:

IF the 'traffic light' is green THEN the action is go

Rule 2:

IF the 'traffic light' is red THEN the action is stop

Rule 3:

IF the 'traffic light' is red THEN the action is go

نلاحظ أن لكلا القاعدتين *2* و *3* نفس الشرط. وبالتالي فيُمكن تطبيق كل منهما في حال تحقق الشرط.

تؤلف هذه القواعد مجموعة تضارب، وعلى محرك الاستدلال أن يُقرر أي من القواعد يجب تطبيقه من هذه المجموعة.

تدعى الطريقة التي تقود إلى اختيار القاعدة الواجب تطبيقها من مجموعة التضارب بحل التضارب *Resolution Conflict*

ستقوم استراتيجية السَلسَلة الأمامية بتطبيق كلتا القاعدتين *2* و *3*.

تُطبق القاعدة *2* أولاً لأن ترتيبها قبل القاعدة *3*، وبالتالي سيتم توليد النتيجة *Stop*. كذلك فإن القاعدة *3* ستُطبق لأن شرطها محقق، وسيتم توليد النتيجة *Go!*

3 - 6 - طرق حل التضارب

- **طبّق القاعدة ذات الأولوية الأكبر *highest priority***

يُمكن في تطبيق بسيط إعطاء الأولويات للقواعد عن طريق ترتيبها في قاعدة المعرفة (حيث ستُطبّق القواعد حسب ترتيبها).

- **طبّق القاعدة الخاصة أكثر *most specific rule***

تُدعى هذه الطريقة أيضاً باستراتيجية المطابقة الأكبر *longest matching strategy*. وتعتمد على فرضية أن القاعدة الأكثر خصوصية تعالج معلومات أكثر من القواعد العامة.

- **طبّق القاعدة التي يستخدم البيانات الأحدث *most recently entered data***

تقوم هذه الطريقة بإضافة معرف الزمن إلى كل حقيقة في قاعدة البيانات. في حال التضارب، نُطّبق القاعدة التي تعتمد شروطها على الحقائق الأحدث.

3 - 7 - تعلم القواعد *Rules Learning*

يوجد العديد من الطرق المقترحة لتعلم القواعد بشكل استنتاجي.

سنشرح واحدةً منها فيما يلي.

ولتوضيح إجرائية التعلم، سنستخدم ثانيةً مثال الموافقة على قرض مصرفي. إلا أنه عوضاً عن البدء بإعطاء القواعد لهذه المسألة، سنفترض أننا أُعطينا مجموعة أمثلة للتعلم تتألف من قيم الحقائق لعدد من العملاء.

لنعتبر مثلاً مجموعة التعلم المعطاة في الجدول التالي. (نستخدم 1 لـ *True* و 0 لـ *False*).

OK	BAL	INC	RATING	APP	
0	1	0	0	1	1
0	0	1	0	0	2
1	1	0	1	1	3
1	1	1	1	0	4
0	0	1	1	0	5
1	0	1	1	1	6
1	1	1	1	1	7
0	0	1	0	1	8
0	0	0	1	1	9

بيانات المصرف

يُمكن الحصول على هذا الجدول بالرجوع إلى سجلات طلبات القروض المصرفية والقرارات التي اتخذها موظفو المصرف بشأن الموافقة على القرض. ندعو الأمثلة التي تأخذ فيها *OK* القيمة *True* بأمثلة موجبة، والتي تأخذ فيها القيمة *False* بأمثلة سالبة. ونريد باستخدام مجموعة التعلم السابقة استنتاج قواعد من الشكل:

$$\alpha_1 \wedge \alpha_2 \wedge \ldots \alpha_n \Rightarrow OK$$

إذا كان لمقدمة قاعدة القيمة *True* من أجل مثال معين من مجموعة التعلم، فإننا نقول بأن القاعدة تُغطي *cover* المثال. يُمكن أن نُغير في أي قاعدة لجعلها تُغطي عدداً أقل من الأمثلة وذلك بإضافة ذرة لمقدمتها. سيجعل مثل هذا التغيير القاعدة أكثر خصوصية *specific*.

يُمكن لقاعدتين أن تُغطيا أمثلة أكثر مما تُغطيه قاعدة واحدة. يجعل إضافة قاعدة النظام الذي يستخدم هذه القواعد أكثر عمومية *General*. سنبحث عن مجموعة من القواعد التي تغطي كل وفقط الأمثلة الموجبة في مجموعة التعلم.

. يُمكن أن تكون عملية البحث عن مثل هذه القواعد مكلفة حسابياً.
نشرح هنا طريقة «شرهه *Greedy*» ندعوها «فرّق تسُد *and Separate Conquer*».

نحاول في هذه الطريقة أولاً أن نجد قاعدة واحدة تُغطي الأمثلة الموجبة فقط (حتى لو كانت لا تُغطي جميع الأمثلة الموجبة)، وذلك بالبدء بقاعدة تُغطي جميع الأمثلة (الموجبة والسالبة)، ثم نجعلها بالتدريج أكثر خصوصية بإضافة ذرات لمقدمتها.

وبما أن قاعدة واحدة قد لا تكفي لتغطية جميع الأمثلة الموجبة، سنقوم بالتدريج بإضافة القواعد حتى الوصول إلى مجموعة كاملة من القواعد التي تُغطي كل وفقط الأمثلة الموجبة.

لنرى تطبيق هذه الطريقة على مثالنا. نبدأ أولاً بالقاعدة المؤقتة التالية والتي تُغطي جميع الأمثلة:

$$T \Rightarrow OK$$

يجب أن نُضيف الآن ذرة لنجعلها تغطي أمثلة سالبة بشكل أقل، وبهدف الوصول لقاعدة تُغطي الأمثلة الموجبة فقط. والسؤال المطروح هنا هو أي من الذرات *{APP, RATING, INC, BAL}* يجب إضافتها؟.

يوجد العديد من المعايير الممكن استخدامها للقيام بالاختيار. ومن هذه المعايير البسيطة النسبة سهلة الحساب التالية:

$$r_\alpha = n^+_\alpha / n_\alpha$$

حيث:

n_α هي العدد الكلي للأمثلة (الموجبة والسالبة) المغطاة من قبل المقدمة (الجديدة) للقاعدة بعد إضافة الذرة α للمقدمة.

n^+_α هو العدد الكلي للأمثلة الموجبة المغطاة من قبل المقدمة (الجديدة) للقاعدة بعد إضافة α لمقدمتها.

نختار الذرة α التي تُعطي أكبر قيمة لـ r_α.

تكون هذه القيم في مثالنا:

$$r_{APP} = 3/6 = 0.5$$

$$r_{RATING} = 4/6 = 0.667$$

$$r_{INC} = 3/6 = 0.5$$

$$r_{BAL} = 3/4 = 0.75$$

ولذا، سنختار الذرة BAL، مما يُعطي القاعدة المؤقتة:

$$BAL \Rightarrow OK$$

تُغطي هذه القاعدة الأمثلة الموجبة 3 و 4 و 7. إلا أنها تُغطي المثال السالب 1، ولذا فيجب أن نُخصصها أكثر.

نستخدم نفس الأسلوب السابق لاختيار ذرة أخرى. وبالطبع، فإن حساب r_α الآن يجب أن يأخذ بالاعتبار أننا قررنا أن مقدمة القاعدة هو BAL. وبالتالي يكون:

$$r_{APP} = 2/3 = 0.667$$

$$r_{RATING} = 3/3 = 1$$

$$r_{INC} = 2/2 = 1$$

لدينا هنا تعادل بين $RATING$ و INC. نختار $RATING$ لأنها تعتمد على مجموعة أكبر من الأمثلة. تُغطي القاعدة الجديدة الأمثلة الموجبة فقط:

$$BAL \wedge RATNG \Rightarrow OK$$

ولذا، فلسنا بحاجة لإضافة ذرات جديدة إلى مقدمة هذه القاعدة.

إلا أن القاعدة:

$BAL \wedge RATNG \Rightarrow OK$

لا تُغطي جميع الأمثلة الموجبة، إذ أنها لا تُغطي المثال الموجب *6*. ولذا فعلينا إضافة قاعدة أخرى.

لتعلم القاعدة التالية، نقوم أولاً بحذف جميع الأمثلة الموجبة المغطاة بالقاعدة الأولى من الجدول للحصول على الجدول التالي:

OK	BAL	INC	RATING	APP	
0	1	0	0	1	1
0	0	1	0	0	2
0	0	1	1	0	5
1	0	1	1	1	6
0	0	1	0	1	8
0	0	0	1	1	9
البيانات الباقية					

ونُعيد الآن تطبيق الطريقة مع الجدول الجديد بدءاً من القاعدة $T \Rightarrow OK$ والتي تُغطي الأمثلة السالبة *1* و *2* و *5* و *8* و *9*.

ولاختيار الذرة الواجب إضافتها للقاعدة نحسب:

$r_{APP} = 1/4 = 0.25$

$r_{RATING} = 0/3 = 0$

$r_{INC} = 1/4 = 0.25$

$r_{BAL} = 0/1 = 0$



يوجد لدينا مرة أخرى تعادل بين *APP* و *INC*.

لنختار بشكل عشوائي *APP* ما يُعطي القاعدة:

$$APP \Rightarrow OK$$

تُغطي هذه القاعدة الأمثلة السالبة *1* و *8* و *9*، ولذا فيجب إضافة ذرة جديدة لمقدمتها. نحسب إذاً النسب:

$$r_{RATING} = 1/2 = 0.5$$

$$r_{INC} = 1/2 = 0.5$$

$$r_{BAL} = 0/1 = 0$$

ونختار *RATING* ما يُعطي القاعدة:

$$APP \wedge RATING \Rightarrow OK$$

تُغطي هذه القاعدة المثال السالب *9*. وبجعل هذه القاعدة أكثر خصوصية (باتباع نفس الطريقة) تنتج القاعدة:

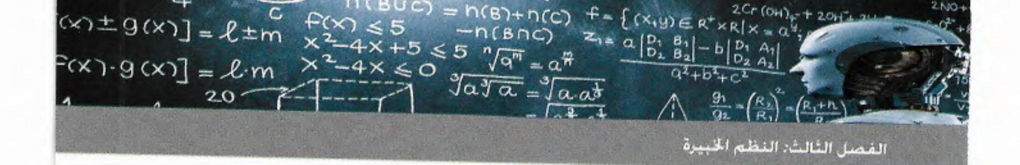
$$APP \wedge RATING \wedge INC \Rightarrow OK$$

تُغطي القواعد المستنتجة:

$$BAL \wedge \ RATNG \Rightarrow OK$$

$$APP \wedge RATING \wedge INC \Rightarrow OK$$

جميع وفقط الأمثلة الموجبة. وبهذا نكون قد انتهينا.



3 - 8 - لغة البرمجة المنطقية Prolog

نقوم عند استخدام *Prolog* بكتابة الحقائق والقواعد الضرورية. يقوم *Prolog* بعدها باستخدام الحاكمة الاستنتاجية لحل المسألة المعينة.

يحوي *Prolog* على محرك استدلال يقوم باستخدام استراتيجية بحث من النمط أعلى-أسفل و يسار-يمين *right-left and down-top* على الحقائق والقواعد لاستكشاف الحلول.

تُستخدم الحقائق لتمثيل المعارف. فمثلاً يُمكن كتابة المعارف التالية:

Bill likes Cindy.

Cindy likes Bill.

Bill likes dogs.

بلغة *Prolog*:

likes(bill, cindy).

likes(cindy, bill).

likes(bill, dogs).

ليكن لدينا مثلاً باللغة الطبيعية:

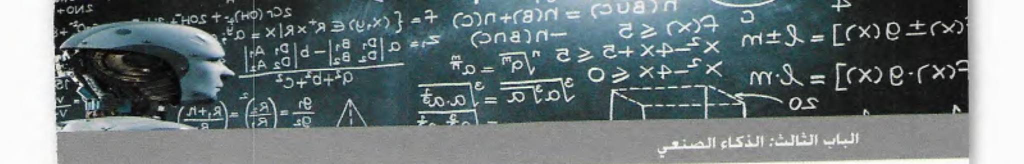
Cindy likes everything that Bill likes.

Caitlin likes everything that is green.

نكتب ذلك في *Prolog*:

likes(cindy, Something):- likes(bill, Something).

likes(caitlin, Something):- green(Something).

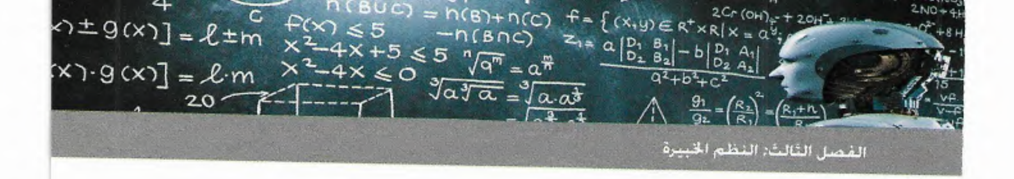


مثال: من يستطيع أن يشتري

```
person(kelly).
person(judy).
person(ellen).
person(mark).
car(lemon).
car(hot_rod).
likes(kelly, hot_rod).
likes(judy, pizza).
likes(ellen, tennis).
likes(mark, tennis).
for_sale(pizza).
for_sale(hot_rod).
can_buy(X,Y):- person(X), car(Y), likes(X,Y), for_sale(Y).
```

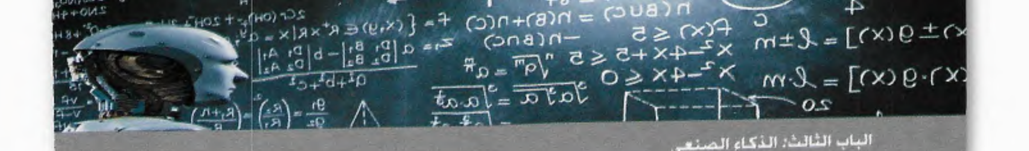
سيُعطي الهدف *(can_buy(X,Y* النتائج التالية:

```
can_buy(judy, pizza)
can_buy(kelly, hot_rod)
```



مثال: ماهي أوراق الشدة التي مجموعها عدد معين

```
card(ace,14).
card(king,13).
card(queen,12).
card(knight,11).
card(10,10).
card(9,9).
card(8,8).
card(7,7).
card(6,6).
card(5,5).
card(4,4).
card(3,3).
card(2,2).
/* find three cards giving the value of N */
find(One, Two, Three, N):-
card(One,Val1),
card(Two,Val2),
card(Three,Val3),
Val1 + Val2 + Val3 = N.
```



مثال: **حساب العاملي لعدد طبيعي**

```
factorial(1,1):-!.
factorial(X,FactX):-
 Y=X-1,
 factorial(Y,FactY),
 FactX = X*FactY.
```

مثال: **الرفع لقوة**

```
p(_,0,1):-!.
p(X,1,X):-!.
p(X,N,Result):-
N1 = N-1,
p(X,N1,XN1),
Result = X * XN1.
```

3 - 9 - استخدام القطع Cut

يلغي استخدام *Cut* نقاط التراجع على يسارها وعن القواعد السابقة المشابهة.

مثال:

```
cutcount2(X) :-
   X >= 0, !,
   console::write('\r',X),
```

```
NewX = X+1,
cutcount2(NewX).
cutcount2(X) :-
console::write(«X is negative.»).
```

3 - 10 - القوائم Lists

أمثلة:

[1, 2, 3, 4]

["toto", "titi", "tata"]

[]

- يُمكن أن تكون عناصر قائمة قوائم
- يُميز رأس القائمة عن بقية العناصر باستخدام |

أمثلة:

- *L=[a,b,c,d,e,f], L=[Head|Tail].*

  Head = a, Tail=[b,c,d,e,f].

- *L=[a,b,c,d,e,f], L=[X,Y|_].*

  X=a, Y=c.

- *L=[a,b,c,d,e,f], L=[_,_|Tail].*

  Tail=[c,d,e,f].



مثال: توليد قائمة من الأعداد الصحيحة

```
genl(0,[]):-!.
genl(N,[N|L]):-
N1=N-1,
genl(N1,L).
```

مثال: عدّ عناصر قائمة

```
length_of([], 0).
length_of([_|T],L):-
   length_of(T,TailLength),
   L = TailLength + 1.
```

مثال: انتماء عنصر لقائمة

```
member(X,[X|_]).
member(X,[_|List]) :-
 member(X,List).
```

مثال: إضافة عنصر لقائمة

```
append([],L,L).
append([H|L1],L2,[H|L3]):-
 append(L1,L2,L3).
```

3 - 11 - عدم التوكيد *Uncertainty*

نستعمل في حياتنا لغة يشوبها الالتباس وتعاني من عدم الدقة. فنحن نصف الحقائق باستخدام تعابير مثل غالباً أو بعض الأحيان أو نادراً. وبالتالي فمن الصعوبة التعبير عن معارفنا باستخدام الشكل *THEN-IF* لقواعد الاستدلال.

3 - 12 - محاكمة بايز *Bayesian Reasoning*

لنفرض أن جميع قواعد الاستدلال في قاعدة المعرفة لها الشكل التالي:

IF H is true THEN E is true with probability p

تُمثّل عادة H في النظم الخبيرة فرضية *hypothesis*، بينما يكون E دليل *evidence* يؤيد هذه الفرضية.

يكون لدينا من أجل مجموعة من الفرضيات *Hm,...,H2,H1* ومجموعة من الأدلة *En,...,E2,E1*:

$$p(H_i|E_1\,E_2\,\ldots E_n)=\frac{p(E_1|H_i)\times p(E_2|H_i)\times\ldots\times p(E_n|H_i)\times p(H_i)}{\sum_{k=1}^{m}p(E_1|H_k)\times p(E_2|H_k)\times\ldots\times p(E_n|H_k)\times p(H_k)}$$

مثال:

ليكن لدينا المسألة التالية:

لنفترض أن خبير أُعطي ثلاثة أدلة مستقلة:

E_1, E_2, E_3

مع ثلاث فرضيات:

H_1, H_2, H_3

مع الاحتمالات الأولية لهذه الفرضيات:

$p(H_1), p(H_2), p(H_3)$

يُحدّد الخبير أيضاً الاحتمالات الشرطية لحدوث كل دليل من أجل كل الفرضيات المحتملة.

يُبين الجدول التالي مختلف الاحتمالات المعطاة من قبل الخبير:

Probability	Hypothesis		
	$i=1$	$i=2$	$i=3$
$p(H_i)$	0.40	0.35	0.25
$p(E_1 \vert H_i)$	0.3	0.8	0.5
$p(E_2 \vert H_i)$	0.9	0.0	0.7
$p(E_3 \vert H_i)$	0.6	0.7	0.9

لنفرض أننا لاحظنا أولاً الدليل E_3، يقوم النظام الخبير بحساب الاحتمالات اللاحقة لكل الفرضيات:

$$p(H_i|E_3) = \frac{p(E_3|H_i) \times p(H_i)}{\sum_{k=1}^{3} p(E_3|H_k) \times p(H_k)}, \qquad i = 1, 2, 3$$

وبهذا يكون:

$$p(H_1|E_3) = \frac{0.6 \cdot 0.40}{0.6 \cdot 0.40 + 0.7 \cdot 0.35 + 0.9 \cdot 0.25} = 0.34$$

$$p(H_2|E_3) = \frac{0.7 \cdot 0.35}{0.6 \cdot 0.40 + 0.7 \cdot 0.35 + 0.9 \cdot 0.25} = 0.34$$

$$p(H_3|E_3) = \frac{0.9 \cdot 0.25}{0.6 \cdot 0.40 + 0.7 \cdot 0.35 + 0.9 \cdot 0.25} = 0.32$$

لاحظ أنه بعد ملاحظة الدليل E_3، فإن الاعتقاد بالفرضية H_1 ينخفض ويصبح مساوياً للاعتقاد بالفرضية H_2. كذلك فإن الاعتقاد بالفرضية H_3 يرتفع ويصبح تقريباً مساوياً للاعتقاد بـ H_1 و H_2.

لنفرض الآن ظهور الدليل E_1. تُصبح الاحتمالات اللاحقة:

$$p(H_i|E_1E_3) = \frac{p(E_1|H_i)\times p(E_3|H_i)\times p(H_i)}{\sum_{k=1}^{3} p(E_1|H_k)\times p(E_3|H_k)\times p(H_k)}, \qquad i = 1, 2, 3$$

وبالتالي:

$$p(H_1|E_1E_3) = \frac{0.3\cdot 0.6\cdot 0.40}{0.3\cdot 0.6\cdot 0.40 + 0.8\cdot 0.7\cdot 0.35 + 0.5\cdot 0.9\cdot 0.25} = 0.19$$

$$p(H_2|E_1E_3) = \frac{0.8\cdot 0.7\cdot 0.35}{0.3\cdot 0.6\cdot 0.40 + 0.8\cdot 0.7\cdot 0.35 + 0.5\cdot 0.9\cdot 0.25} = 0.52$$

$$p(H_3|E_1E_3) = \frac{0.5\cdot 0.9\cdot 0.25}{0.3\cdot 0.6\cdot 0.40 + 0.8\cdot 0.7\cdot 0.35 + 0.5\cdot 0.9\cdot 0.25} = 0.29$$

لاحظ أن الفرضية H_2 أصبحت الآن هي المرجحة.

وبعد ظهور الدليل E_2، تُصبح الاحتمالات اللاحقة النهائية لجميع الفرضيات:

$$p(H_i|E_1E_2E_3) = \frac{p(E_1|H_i)\times p(E_2|H_i)\times p(E_3|H_i)\times p(H_i)}{\sum_{k=1}^{3} p(E_1|H_i)\times p(E_2|H_i)\times p(E_3|H_i)\times p(H_k)}, \qquad i = 1, 2, 3$$

وبالتالي:

$$p(H_1|E_1E_2E_3) = \frac{0.3\cdot 0.9\cdot 0.6\cdot 0.40}{0.3\cdot 0.9\cdot 0.6\cdot 0.40 + 0.8\cdot 0.0\cdot 0.7\cdot 0.35 + 0.5\cdot 0.7\cdot 0.9\cdot 0.25} = 0.45$$

$$p(H_2|E_1E_2E_3) = \frac{0.8\cdot 0.0\cdot 0.7\cdot 0.35}{0.3\cdot 0.9\cdot 0.6\cdot 0.40 + 0.8\cdot 0.0\cdot 0.7\cdot 0.35 + 0.5\cdot 0.7\cdot 0.9\cdot 0.25} = 0$$

$$p(H_3|E_1E_2E_3) = \frac{0.5\cdot 0.7\cdot 0.9\cdot 0.25}{0.3\cdot 0.9\cdot 0.6\cdot 0.40 + 0.8\cdot 0.0\cdot 0.7\cdot 0.35 + 0.5\cdot 0.7\cdot 0.9\cdot 0.25} = 0.55$$

نلاحظ أن الترتيب الأولي للفرضيات كان:

H_1, H_2, H_3

إلا أنه، وبعد ظهور الأدلة، تبقى فقط الفرضيتان H_1 و H_3 ممكنتين.

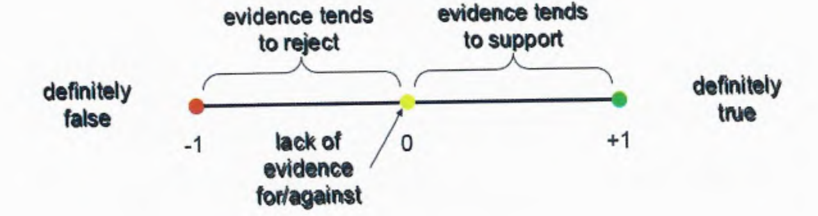
3 - 13 - نظرية معامل الثقة Certainty Factors Theory

تُمثل نظرية معامل الثقة بديلاً مهماً ومنتشراً لطرق محاكمة بايز.

يقيس معامل الثقة *factor certainty* لفرضية *H* درجة اعتقاد الخبير *.belief s'expert*

تتراوح قيمة معامل الثقة بين *1* (مؤكد صحيح) وبين *-1* (مؤكد خطأ).

- *CF=1* الفرضية صحيحة تماماً
- *CF=-1* الفرضية خاطئة تماماً
- *CF=0* لا يوجد دليل مع-ضد الفرضية
- *CF>0* يوجد دلائل تدعم صحة الفرضية
- *CF<0* لا يوجد دلائل تدعم صحة الفرضية



تتألف قاعدة المعرفة في النظم الخبيرة التي تعتمد على معامل الثقة من قواعد استدلال لها الشكل التالي:

IF E THEN H {cf }

حيث يُمثل معامل الثقة *cf* الاعتقاد بالفرضية *H* إذا علمنا وقوع الدليل *E*.

3 - 14 - حساب معامل الثقة

يجب حساب ثقة نتيجة القاعدة عندما يكون الدليل في مقدمة القاعدة غير مؤكد.

$$cf(H,E) = cf(E) * cf$$

مثلاً، إذا كان لدينا القاعدة:

IF sky is clear THEN the forecast is sunny {cf 0.8}

وكانت قيمة معامل الثقة لـ *"sky is clear"* هي *0.5* فإن:

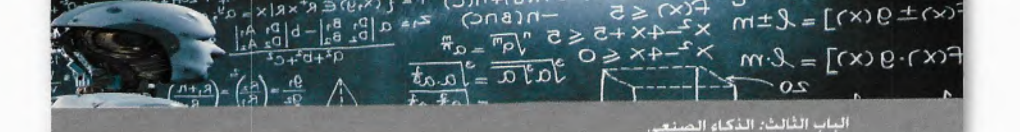
$$cf(H,E) = 0.5 * 0.8 = 0.4$$

حالة قواعد الربط

IF <evidence E_1>
⋮
AND <evidence E_n>
THEN <hypothesis H> {cf}

يُحسب معامل الثقة لفرضية *H* كما يلي:

$$cf(H, E1 \wedge E2 \wedge \ldots \wedge En = (\min\{cf(E1), cf(E2), \ldots, cf(En)\} * cf$$



مثال:

IF sky is clear

AND the forecast is sunny

THEN the action is 'wear sunglasses' cf 0.8

وإذا كان معامل الثقة لـ «*sky is clear*" هو *0.9* ومعامل الثقة لـ «*the forecast is sunny*" هو *0.7* يكون:

$$cf\,(H, E_1 \wedge E_2) = \min\,\{0.9,\ 0.7\} * 0.8 = 0.56$$

حالة قواعد الفصل

IF <evidence E_1>
⋮
OR <evidence E_n>
THEN <hypothesis H> $\{cf\}$

يُحسب معامل الثقة لفرضية H كما يلي:

$$cf\,(H, E_1 \text{Ú} E_2 \text{Ú} \dots \text{Ú} E_n) = \max\,\{cf\,(E_1),\ cf\,(E_2), \dots,\ cf\,(E_n)\} * cf$$

مثال:

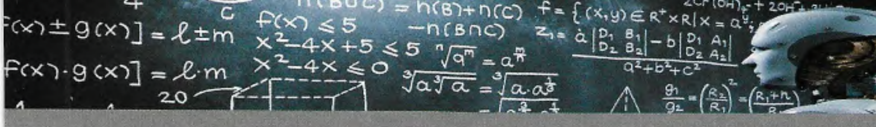
IF sky is overcast

OR the forecast is rain

THEN the action is 'take an umbrella' {cf 0.9}

وإذا كان معامل الثقة لـ «*sky is clear*" هو *0.9* ومعامل الثقة لـ «*the forecast is sunny*" هو *0.7* يكون:

$$cf\,(H, E_1 \text{Ú} E_2) = \max\,\{0.9,\ 0.7\} * 0.9 = 0.81$$



دمج معاملات القواعد

عندما نحصل على نفس النتيجة من أكثر من قاعدة، فيجب القيام بدمج معاملات الثقة التي نحصل عليها جراء تطبيق كل قاعدة بطريقة معينة للوصول إلى معامل الثقة النهائي للنتيجة.

لنفرض مثلاً أن لدينا القاعدتين التاليتين في قاعدة المعرفة:

Rule 1: IF A is X THEN C is Z {cf 0.8}

Rule 2: IF B is Y THEN C is Z {cf 0.6}

والسؤال: ما هو معامل الثقة لـ Z إذا كانت كلتا القاعدتين قابلتين للتطبيق؟

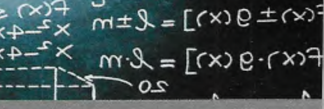
دمج معاملات القواعد

لحساب عامل الثقة الناتج عن أكثر من قاعدة، نستخدم:

$$cf(cf_1, cf_2) = \begin{cases} cf_1 + cf_2 \times (1 - cf_1) & \text{if } cf_1 > 0 \text{ and } cf_2 > 0 \\ \dfrac{cf_1 + cf_2}{1 - min\,[|cf_1|, |cf_1|]} & \text{if } cf_1 < 0 \text{ or } cf_2 < 0 \\ cf_1 + cf_2 \times (1 + cf_1) & \text{if } cf_1 < 0 \text{ and } cf_2 < 0 \end{cases}$$

حيث:

- cf_1 هو معامل الثقة بالفرضية H والناتج عن القاعدة $Rule_1$
- cf_2 هو معامل الثقة بالفرضية H والناتج عن القاعدة $Rule_2$
- $|cf_1|$ و $|cf_2|$ هي القيم المطلقة لـ cf_1 و cf_2.



مثال:

ليكن لدينا:

IF A and B or C and not D

THEN X with CF 0.6

ومعاملات الثقة:

CF(A) = 0.3

CF(B) = 0.5

CF(C) = 0.4

CF(D) = -0.7

والمطلوب حساب معامل الثقة للنتيجة X.

CF(E) =CF (A and B or C and not D)

=max(min(0.3,0.5),min(0.4,0.7))

=max(0.3,0.4) =0.4

CF(H,E)=0.6

CF(X)=CF(H) =CF(E) * CF(H,E) =0.4 * 0.6 =0.24

مثال:

ليكن لدينا:

Rule 1: IF A OR B THEN C (certainty factor 0.3)

Rule 2: IF C OR D THEN H (certainty factor 0.8)

Rule 3: IF E OR F THEN H (certainty factor 0.2)

مع معاملات الثقة:

A is 0.2, B is 0.5, D is 0.3, E is 0.6, F is 0.7

والمطلوب حساب معامل الثقة لـ *H*.

نقوم بالحل كما يلي:

IF A OR B , so the max value of A and B: max (0.2, 0.5) = 0.5

then the certainty of C = 0.5 * certainty factor 0.3 = 0.15

IF C OR D, so the max value of C and D: max (0.15, 0.3) = 0.3

then the certainty of H = 0.3 * certainty factor 0.8 = 0.24.

IF E OR F, so the max value of E and F: max (0.6, 0.7) = 0.7

then the certainty of H = 0.7 * certainty factor 0.2 = 0.14

وبالتالي يكون معامل الثقة:

$$0.24 + 0.14 (1 - 0.24) = 0.3464$$

3 - 15 - أسئلة متعددة الخيارات

1. لتكن مجموعة القواعد التالية واحتمالاتها:

R1: If Battery is bad Then horn does not work with prob=0.3

R2: If Battery is bad Then engine does not start with prob=0.6

R3: If Spark plugs are bad Then horn does not work with prob=0.6

R4: If Spark plugs are bad Then engine does not start with prob=0.4

مع الاحتمالات:

prob(Battery is bad)= 0.4

prob(Spark plugs are bad)= 0.6

وبفرض أن لدينا المشاهدات التالية:

"horn does not work"

"engine does not start"

احسب احتمال الفرضية التالية:

"Spark plugs are bad"

a. 0.33

b. 0.66

c. 0

d. ولا خيار مما سبق



2. ليكن لدينا القواعد التالية في *Prolog*

```
A([X|_], 0, X).
A([_|T], I, X) :- (I > 0) , A(T, I - 1, X).
B(_, [ ], [ ]).
B(L, [I|T], [X|R]) :- A(L, I, X), B(L, T, R).
```

تكون قيمة *R* بعد الاستدعاء التالي:

```
B([1,3,5,7], [0,3], R)
```

a. **R = [1,7]**

b. R = [1,6]

c. R = [2,7]

d. غير ذلك

3. لتكن مجموعة القواعد التالية:

Rule 1 : IF A OR B THEN C (certainty factor 0.3)

Rule 2 : IF C OR D THEN H (certainty factor -0.8)

Rule 3 : IF E OR F THEN H (certainty factor 0.2)

مع معاملات الثقة:

$CF(A) = 0.2$

$CF(B) = 0.5$

$CF(D) = 0.3$

$CF(E) = -0.6$

$CF(F) = -0.7$

يكون معامل الثقة لـ H مساوياً إلى:

a. 0.66

b. -0.66

c. 0

d. **<u>ولا خيار مما سبق</u>**

4. لتكن مجموعة الأمثلة التالية:

	A	B	C	D	OK
1	0	1	0	1	0
2	0	0	1	0	0
3	1	1	0	1	1
4	1	0	1	1	1
5	1	0	1	0	0
6	1	1	1	0	1
7	1	1	1	1	1
8	0	1	1	0	0
9	1	1	0	0	0

تعطي خوارزمية التعلم كقاعدة أولى:

a. $A \wedge B \Rightarrow OK$

b. **<u>$A \wedge D \Rightarrow OK$</u>**

c. $B \wedge C \Rightarrow OK$

d. $B \wedge D \Rightarrow OK$

5. لتكن مجموعة القواعد التالية:

Rule 1 : IF A OR B THEN C (certainty factor 0.3)

Rule 2 : IF C OR D THEN H (certainty factor 0.8)

Rule 3 : IF E OR F THEN H (certainty factor 0.2)

مع معاملات الثقة:

$CF(A) = 0.2$

$CF(B) = 0.5$

$CF(D) = 0.3$

$CF(E) = -0.6$

$CF(F) = -0.7$

يكون معامل الثقة لـ H مساوياً إلى:

a. 0.66

b. **<u>0.13</u>**

c. 0.33

d. -0.13

6. ليكن برنامج *Prolog* التالي:

```
x(0,[ ]).
x(N,[N|L]):- x(N-2,L).
```

يطبع الاستدعاء:

```
x(10, L)
console::write(L)
```

a. *[2 ,6,4 ,8 ,10]*

b. *Stack Overflow* ومن ثم القائمة *[2 ,6,4 ,8 ,10]*

c. ***[2 ,6,4 ,8 ,10]*** **ومن ثم الرسالة** ***Stack Overflow***

d. ولا خيار مما سبق

7. لتكن مجموعة القواعد التالية:

Rule 1 : IF A OR B THEN C (certainty factor 0.3)

Rule 2 : IF C OR D THEN H (certainty factor 0.8)

Rule 3 : IF E OR F THEN H (certainty factor 0.2)

مع معاملات الثقة:

$CF(A) = 0.2$

$CF(B) = 0.5$

$CF(D) = 0.3$

$CF(E) = -0.6$

$CF(F) = -0.7$

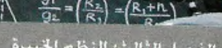
يكون معامل الثقة لـ *H* مساوياً إلى:

a. 0.66

b. -0.13

c. 0.33

d. **0.13**



8. لتكن مجموعة القواعد التالية:

$R1: \quad A \rightarrow B$

$R2: \quad A \rightarrow C$

$R3: \quad B \rightarrow C$

$R4: \quad C \rightarrow D$

$R5: \quad D \rightarrow E$

$R6: \quad G \rightarrow F$

$R7: \quad E \rightarrow F$

$R8: \quad H \rightarrow G$

وحيث الحقيقة الوحيدة الصحيحة هي A. والهدف هو F.

في استراتيجية السلسلة الأمامية تكون عدد الحلقات *cycles* للوصول للهدف:

a. 3

b. <u>4</u>

c. 5

d. 6

9. لتكن مجموعة القواعد التالية:

$R1:\quad A \rightarrow B$

$R2:\quad A \rightarrow C$

$R3:\quad B \rightarrow C$

$R4:\quad C \rightarrow D$

$R5:\quad D \rightarrow E$

$R6:\quad G \rightarrow F$

$R7:\quad E \rightarrow F$

$R8:\quad H \rightarrow G$

وحيث الحقيقة الوحيدة الصحيحة هي A. والهدف هو F.

في استراتيجية السلسلة الخلفية يكون عدد القواعد المنفذة *Fired* للوصول للهدف:

a. 3

b. 4

c. 5

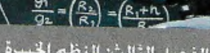
d. 6

10. لتكن لدينا مجموعة القواعد التالية مع الاحتمالات الموافقة:

Let the following rules:

R1:If Battery is bad *Then engine does not start*
prob=0.3

R2:If Battery is bad *Then lights do not come on*
prob=0.8



R3:If Battery is bad prob=0.5	*Then horn does not work*
R4:If Spark plugs are bad prob=0.9	*Then engine does not start*
R5:If Spark plugs are bad prob=0.0	*Then lights do not come on*
R6:If Spark plugs are bad prob=0.7	*Then horn does not work*
R7:If Electricity cable is cut prob=0.6	*Then engine does not start*
R8:If Electricity cable is cut prob=0.7	*Then lights do not come on*
R9:If Electricity cable is cut prob=0.9	*Then horn does not work*

علما أن:

prob(Battery is bad)=0.4 , prob(Spark plugs are bad)=0.35, prob(Electricity cable is cut)=0.25

وبفرض أن لدينا المشاهدات التالية:

"horn does not work", "engine does not start", and "lights do not come on".

احسب احتمال الفرضية:

"Electricity cable is cut".

a. <u>55%</u>

b. 45%

c. 0%

d. ولاخيار مما سبق

11. لتكن لدينا مجموعة القواعد التالية مع معاملات التوكيد:

R1: If engine does not start Then Battery is bad
CF=0.3

R2: If lights do not come on Then Battery is bad
CF=0.8

R3: If horn does not work Then Battery is bad
CF=0.5

وبفرض أن لدينا معاملات التوكيد التالية للمشاهدات:

- CF("horn does not work")= 1
- CF("engine does not start")= -1
- CF("lights do not come on")= -1

احسب معامل التوكيد للفرضية:

CF("Battery is bad")

a. <u>-0.72</u>

b. 0.72

c. -0.27

d. ولاخيار مما سبق

12. ليكن لدينا القواعد التالية في *Prolog*

```
fis(L, X, C) :- fi(L, 0, X, C).
fi([ ], _, _, [ ]).
fi([X|T], I, X, [I|C]) :- fi(T, I + 1 , X, C).
fi([H|T], I, X, C) :- (X != H), fi(T, I + 1, X, C).
```

تكون قيمة *C* بعد الاستدعاء التالي:

```
fis([1,5,5,1,3,5], 5, C)
```

a. <u>C = [1,2,5]</u>

b. C = [2,1,5]

c. C = [5,2,1]

d. غير ذلك

الفصل الرابع

البحث *Search*

يلعب البحث دوراً أساسياً في الكثير من مسائل وتطبيقات الذكاء الصنعي. تُعتبر خوارزميات البحث العمود الفقري لجميع المسائل التي تحتاج لاستكشاف الخيارات المتعددة بشكل منهجي.

يرتبط مفهوم حل المسائل في الذكاء الصنعي ارتباطاً وثيقاً بتقانات البحث الموجه في البيان (سنرى أن البحث الموجه يعني وجود آلية "ذكية" تقود عملية سبر البيان).

4 - 1 - البيانات *Graphs*

- تتواجد البيانات *Graphs* في كل مكان: شبكات الطرق، الخطوط الجوية، شبكات الحواسيب.
- نهتم في جميع الحالات بإيجاد مسار *Path* عبر البيان يحقق خصائص معينة.
- يُمكن أن نكتفي في بعض الحالات بأي مسار، أو نريد إيجاد المسار ذي الكلفة الأقل.
- من أهم منهجيات حل المسائل في الذكاء الصنعي تحويل المسألة إلى مسألة بحث في بيان.

- لاستخدام هذه المنهجية يجب تحديد الحالات والأفعال واختبار الهدف.
- نفترض أن الحالة كاملة *Complete* أي أنها تُمثل جميع جوانب المسألة.
- نفترض أن الأفعال حتمية *Deterministic* أي أننا نعرف الحالة تماماً بعد تطبيق فعل.

4 - 2 - خوارزميات البحث في بيان الحالات

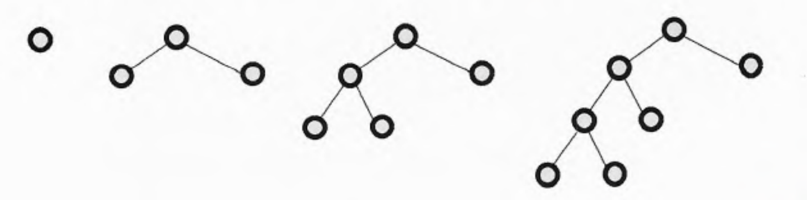
- بعد تعريف فضاء المسألة (*تمثيل الحالات، الأفعال أو المعاملات، الحالة الابتدائية، الحالة النهائية*) يبدأ البحث!
- انطلاقاً من الحالة الابتدائية نعاود تطبيق الأفعال الممكنة حتى الوصول إلى الحالة النهائية.
- إلا أن فضاء البحث عادةً ضخم جداً جداً !
- وبالتالي نحتاج إلى منهجيات تقود البحث.

4 - 3 - خوارزميات البحث الأعمى *Blind Search*

لا تستخدم هذه الخوارزميات معلومات متعلقة بالمسألة.

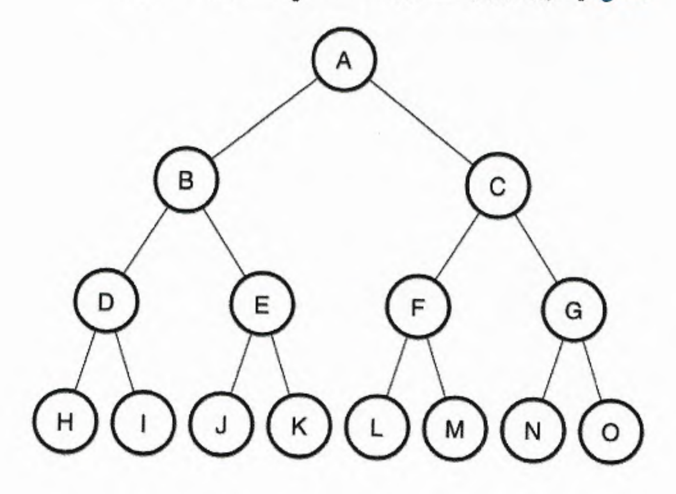
4 - 4 - البحث بالعمق - أولاً *Depth-First Search*

تعتمد هذه الخوارزمية على المبدأ: طوّر العقدة الأعمق غير المطورة.



توضع العقد الحدود في مكدّس من النوع *LIFO* أي الداخل أخيراً الخارج أولاً.

مثال: يكون ترتيب تطوير العقد في الشجرة التالية:

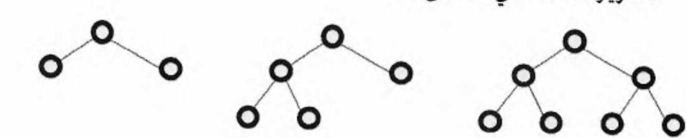


هو:

A, B, D, H, I, E, J, K, C, F, L, M, G, N, O

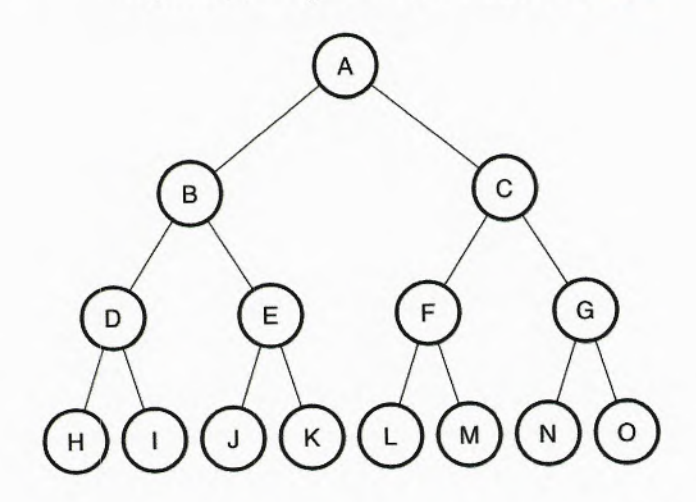
4 - 5 - البحث بالعرض - أولاً *Breadth-First Search*

تعتمد هذه الخوارزمية على المبدأ: طوّر جميع العقد في العمق *i* قبل تطوير العقد في العمق *i+1*.



توضع العقد الحدود في رتل من النوع *FIFO* أي الداخل أولاً الخارج أولاً.

مثال: يكون ترتيب تطوير العقد في الشجرة التالية:

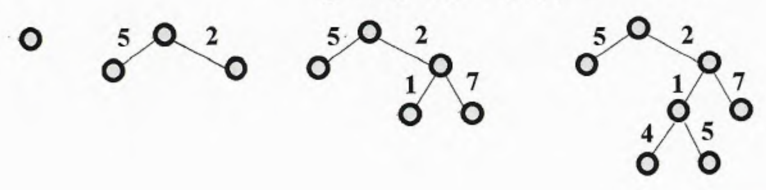


هو:

A ,B ,C ,D ,E ,F ,G ,H ,I ,J ,K ,L ,M ,N ,O

4 - 6 - البحث وفق الكلفة المنتظمة *Uniform-Cost Search*

تعتمد هذه الخوارزمية على المبدأ: طوّر العقدة ذات الكلفة الأقل. أي أن الخوارزمية تقوم بتطوير العقد وفق الكلف المتزايدة. وبالتالي فإن أي عقدة هدف نصل إليها تكون الحل الأمثل.



توضع إذاً العقد الحدود في رتل مرتب تصاعدياً وفق الكلفة. أي أن إضافة العقد الجديدة المولدة تتمّ وبحيث نحافظ على الترتيب التصاعدي للكلف.

4 - 7 - التجريبيات *Heuristics*

إن طرائق البحث العمياء هي طرائق شاملة تهدف إلى إيجاد طريق حل. إلا أن تطبيقها غير واقعي، من أجل معظم المسائل، لأنه يتطلب معالجة عدد هائل من العقد قبل مصادفة الحل.

تكمن الفكرة الأساسية لخوارزميات البحث المطّلعة مع بيان الحل في استعمال معلومة تجريبية خاصة بالمسألة المعالجة تؤدي إلى تخفيض عدد العقد المعالجة.

التجريبية هي تابع عندما يُطبّق على حالة يُعيد رقم يُقدّر قرب هذه الحالة من الهدف.

أي أن التجريبية تُخبر تقريباً كم تبقى للوصول للهدف *(الأرقم الأصغر أفضل)*.

يُمكن أن تبخس التجريبية *underestimate* أو تغالي *overestimate* في تقدير البعد عن الهدف.

سوف نرى أن التجريبيات التي نعتبرها مقبولة *admissible* هي التجريبيات التي يكون تقديرها دائماً أصغر من الكلفة الحقيقية.

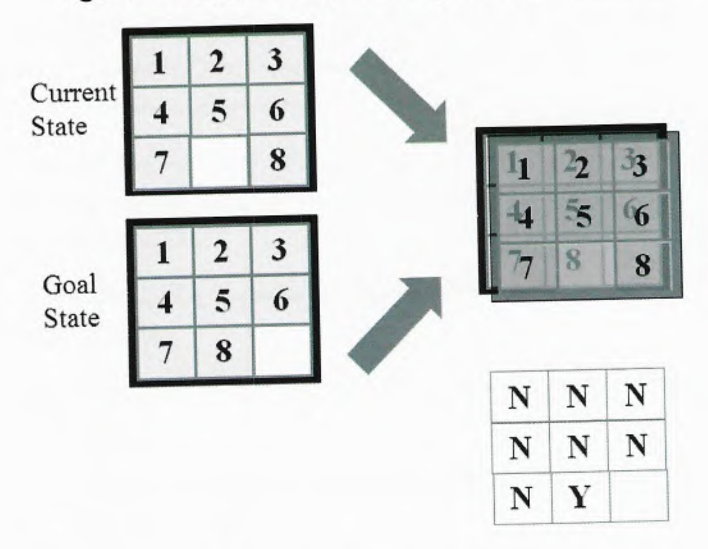
مثال*1*: أقصر طريق

يُمكن في مسألة البحث عن المسار الأمثلي لخريطة من المدن اعتماد مسافة خط النظر بين مدينة ما والمدينة الهدف كمقياس لقرب هذه المدينة من الهدف.

مثال2: لعبة التكوتان

يُمكن في مسألة التاكوتان اعتماد التجريبية التالية لقياس قرب حالة من الهدف:

عدد الخانات الموجودة في غير مكانها الصحيح (ماعدا الفراغ).



في مثالنا تكون قيمة التجريبية *1* إذ فقط لا توجد الخانة *8* في موضعها الصحيح.

مثال3: لعبة التكوتان

يُمكن في مسألة التاكوتان مسافة مانهاتن *Distance Manhattan* لقياس قرب حالة من الهدف:

(مجموع عدد الخانات التي يجب أن نعبرها لنوصل كل خانة إلى مكانها الصحيح).

Current State

3	2	8
4	5	6
7	1	

Goal State

1	2	3
4	5	6
7	8	

3	➡	<u>3</u>

2 spaces

	⬅	8
	⬇	
	<u>8</u>	

3 spaces

<u>1</u>	⬅	
	⬆	
	1	

3 spaces

Total 8

4 - 8 - خوارزمية تسلق التلة *Hill Climbing Search*

ليكن التابع h تقديراً للكلفة من العقدة n إلى الهدف.

ندعو هذا التابع بالتجريبية *heuristic*.

مثلاً، يُمكن في مسألة التنقل بين المدن أن يكون هذا التابع مسافة خط النظر بين المدينة الحالية والمدينة الهدف.

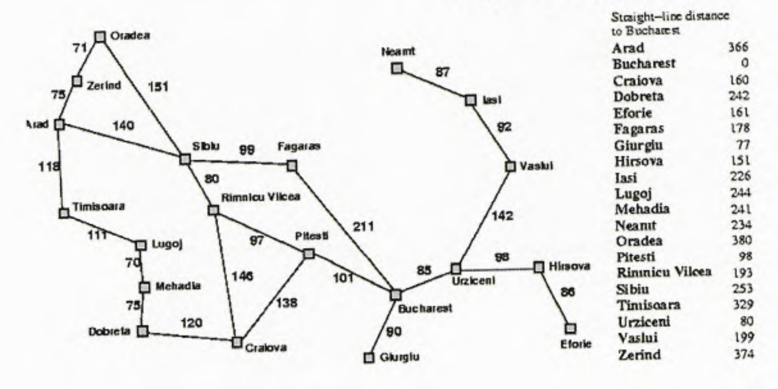
$h_{SLD}(n)$ = *straight-line distance from n to goal*

تقوم خوارزمية تسلق التلة بتطوير العقدة التي يكون تقديرها أقرب للهدف.

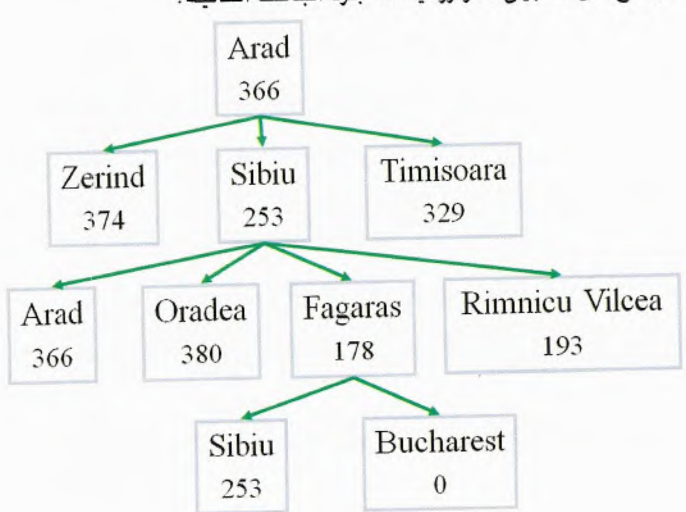
(بخلاف خوارزمية البحث وفق الكلفة المنتظمة والتي تطور العقدة ذات الكلفة الأقل).

مثال: الانتقال من *Arad* إلى *Bucharest*

نستخدم في هذا المثال مسافة خط النظر كتقدير للكلفة المتبقية من مدينة إلى المدينة الهدف.



ينتج عن تطبيق الخوارزمية شجرة البحث التالية:



مثال: لعبة التكوتان

ينتج عن تطبيق خوارزمية تسلق التلة مع اعتماد مسافة مانهاتن شجرة البحث التالية:

(لاحظ سرعة الوصول إلى الحل)

Current State

3	2	8
4	5	6
7	1	

Goal State

1	2	3
4	5	6
7	8	

3	➡	3

2 spaces

	⬅	8
	⬇	
	8	

3 spaces

1	⬅	
	⬆	
	1	

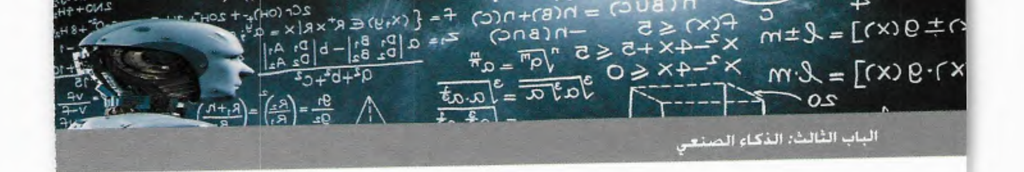
3 spaces

Total 8

4 - 9 - خوارزمية *A

تقوم هذه الخوارزمية بدمج كل من الخوارزميتين: البحث وفق الكلفة المنتظمة وخوارزمية تسلق التلة للوصول إلى خوارزمية كاملة وأمثليه وسريعة. نستخدم في هذه الخوارزميتين التابعين التاليين:

- g كلفة العقدة الحالية
- h تقدير كلفة الوصول إلى الهدف اعتباراً من العقدة الحالية



ونستخدم المحصلة:

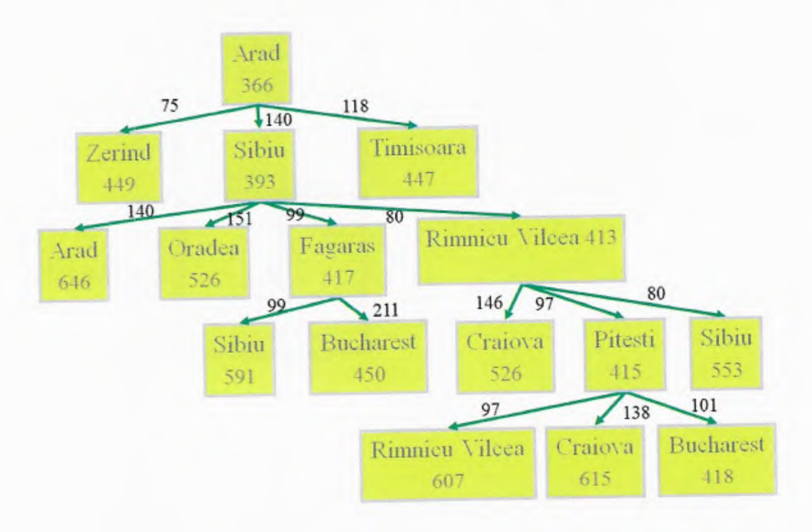
$$f(n) = g(n) + h(n)$$

لترتيب العقد الحدود وفق قيمة التابع f بشكل تصاعدي.

تكون هذه الخوارزمية أمثلية إذا كانت تقدير الكلفة للوصول إلى الهدف دائماً أصغر أو يساوي الكلفة الحقيقية.

مثال: الانتقال من *Arad* إلى *Bucharest*

ينتج عن تطبيق الخوارزمية شجرة البحث التالية:



4 - 10 - مسألة البائع الجوال

بفرض أن لدينا مجموعة من العقد. نُعرّف المسافات بين هذه العقد.
والمطلوب:

إيجاد مسار ذو كلفة أصغرية وبحيث نزور كل عقدة مرة واحدة.

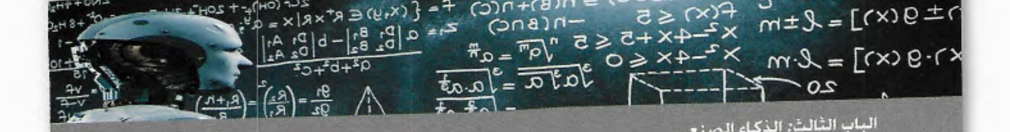




مثال:

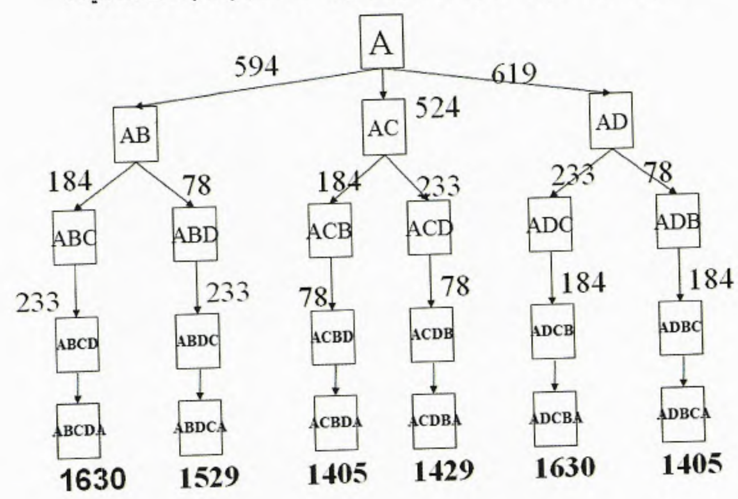
نريد زيارة كل من المدن الخمس التالية وبكلفة أصغرية.

	Aberdeen	Brighton	Cardiff	Dover	Edinburgh
Aberdeen	0	594	524	619	127
Brighton	594	0	184	78	467
Cardiff	524	184	0	233	395
Dover	619	78	233	0	493
Edinburgh	127	467	395	493	0



وذلك بتطبيق الخوارزمية *A.

ينتج عن تطبيق الخوارزمية من أجل $N=4$ شجرة البحث التالية:



لاحظ أن عدد المسارات هو:

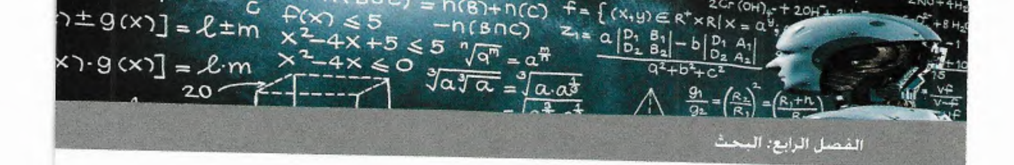
No of paths =(1- n)!

أي مثلاً:

n ,4=p6=

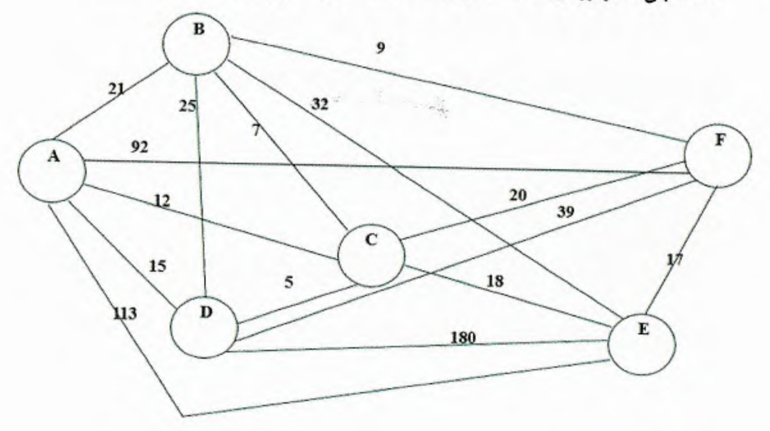
n=5, p=24

n=10, p=362,880



مثال:

لنطبق الخوارزمية A^* على المسألة التالية مع تجريبيات مختلفة:



التجريبيات

لنستخدم التجريبيات التالية:

$H1=0$

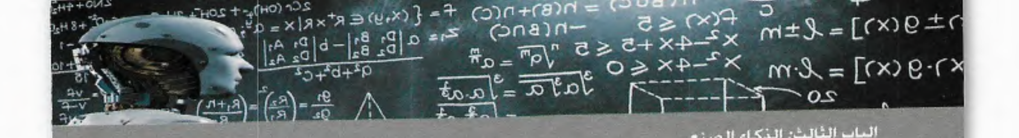
التجريبية 1 = صفر

*H2 = No missed arcs * cost of minimal arc*

التجريبية 2 = عدد الأقواس الناقصة * كلفة أصغر قوس

H3 = Sum of p most short arcs) p is No missed arcs(

التجريبية 3 = مجموع أقصر p قوس (حيث p هو عدد الأقواس الناقصة)



H4 = Sum of p most short arcs connected to nodes that remained to be exited from

Ex: For H4(ABD)=36 Since we have to exit from D,C,E and F

التجريبية4 = مجموع أقصر *p* قوس مرتبط بالعقد المتبقي الخروج منها

H5 = Sum of p most short arcs connected to nodes that remained to be visited

Ex: For H4(ABD)=43 Since we have to visit from C ,E ,F and A.

التجريبية5 = مجموع أقصر *p* قوس مرتبط بالعقد الواجب الدخول فيها

تُعطي جميع التجريبيات السابقة الحل الأمثل إذ أن تقدير الكلفة في كل منها أصغر من الكلفة الحقيقية.

FBADCEF or FECDABF

Cost: 85

يُبين الجدول التالي عدد العقد المولدة وعدد العقد المطورة من أجل كل تجريبية:

Heuristic	*Developed Nodes*	*Created Nodes*
H1	*88*	*159*
H2	*72*	*138*
H3	*62*	*115*
H4	*47*	*97*
H5	*69*	*133*

في حال استخدام التجريبية التالية والتي لاتحقق شرط الأمثلية:

*H6: No of missed arcs * cost of average arc*

التجريبية6 = عدد الأقواس الناقصة * الكلفة الوسطية للأقواس

نحصل على الحل:

FEADCBF

Cost: 166

مع الكلف التالية:

Heuristic	Developed Nodes	Created Nodes
H6	9	22

وفي حال استخدام التجريبية التالية والتي لاتحقق شرط الأمثلية:

*H7: No of missed arcs * cost of max arc*

التجريبية7 = عدد الأقواس الناقصة * كلفة أكبر قوس

نحصل على الحل:

FBCDAEF

Cost: 166

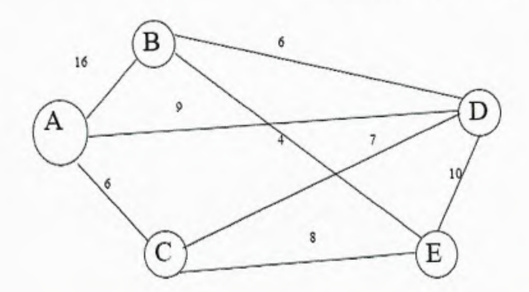
مع الكلف التالية:

Heuristic	Developed Nodes	Created Nodes
H7	13	17

4 - 11 - تمارين محلولة

1. ليكن البيان التالي وحيث المطلوب الانتقال من A إلى E.

ماذا سيكون الناتج باعتماد كل من الخوارزميات التالية: العمق أولاً، العرض أولاً، الكلفة المنتظمة.



الحل:

العمق أولاً	A, B, D, C, E
العرض أولاً	A, B, E
الكلفة المنتظمة	A, C, E

2. ماذا سيكون الناتج باستخدام خوارزمية تسلق التلة وخوارزمية A^* علماً بأن لدينا تقدير للمسافة بين كل عقدة والعقدة E كما يلي:

A	B	C	D	E
10	2	8	5	0

الحل:

تسلق التلة	A, B, E
A^*	A, C, E

4 - 12 - أسئلة متعددة الخيارات

3. تعطي خوارزمية البحث من النمط العمق أولاً *Depth-First Search*:

a. حل أمثلي دائماً فقط في مسألة البائع الجوال.

b. حل أمثلي دائماً فقط في مسائل الألعاب بلاعب واحد (مثل *8-puzzle*).

c. حل أمثلي دائماً في جميع مسائل البحث.

***d*. ولا خيار من الخيارات الثلاث السابقة.**

4. تعطي خوارزمية البحث من النمط العرض أولاً *Breadth-First Search*:

a. حل أمثلي دائماً فقط في مسألة البائع الجوال.

***b*. حل أمثلي دائماً فقط في مسائل الألعاب بلاعب واحد (مثل *8-puzzle*).**

c. حل أمثلي دائماً في جميع مسائل البحث.

d. ولا خيار من الخيارات الثلاث السابقة.

5. تعطي خوارزمية البحث من النمط البحث المنتظم *Uniform Search*:

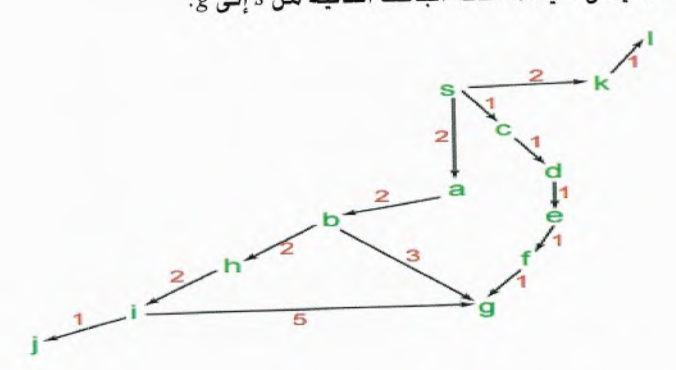
a. حل أمثلي دائماً فقط في مسألة البائع الجوال.

b. حل أمثلي دائماً فقط في مسائل الألعاب بلاعب واحد (مثل *8-puzzle*).

***c*. حل أمثلي دائماً في جميع مسائل البحث.**

d. ولا خيار من الخيارات الثلاث السابقة.

6. ليكن لدينا مسألة البحث التالية من *s* إلى *g*:



مع قيمة التجريبية كما يلي:

h(a,2), h(b,3), h(c,4), h(d,3), h(e,2), h(f,1), h(g,0), h(h,4), h(i,5), h(j,6), h(k,5), h(l,6), h(s,4).

تكون كلفة المسار الناتج بتطبيق خوارزمية تسلق التلة *Hill Climbing*

a. 5

b. <u>7</u>

c. 13

d. غير ذلك

ليكن لدينا اللعبة البسيطة التالية *(4-puzzle)* :

وليكن لدينا التجريبيتين التاليتين:

Goal

1	2
-	3

Start

-	1
3	2

$h1$ مسافة مانهاتن لوضع الفراغ في مكانه.

$h2$ عدد الخانات الموجودة في غير مكانها الصحيح (لا نحسب الفراغ).

مثلاً للحالة الابتدائية يكون: $h1=1, h2=3$

7. يكون عدد الحالات المختلفة التي سيتم توليدها حتى الوصول للهدف (بما فيها الحالة الابتدائية والنهائية) بتطبيق الخوارزمية *A مع التجريبية $h1$:

a. 7

b. 6

c. 5

d. ولا خيار مما سبق

8. يكون عدد الحالات المختلفة التي سيتم توليدها حتى الوصول للهدف (بما فيها الحالة الابتدائية والنهائية) بتطبيق الخوارزمية *A مع التجريبية $h2$:

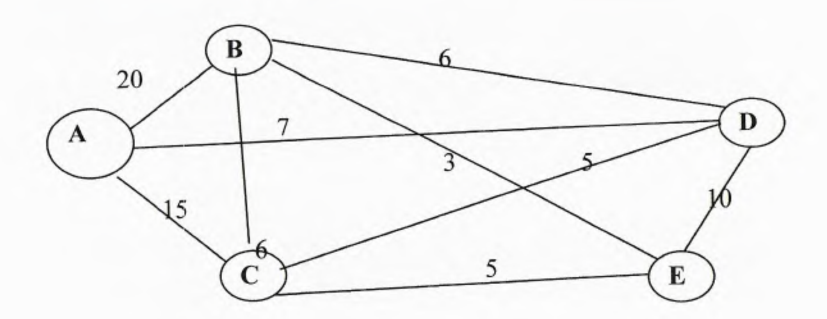
a. 4

b. 5

c. 6

d. ولا خيار مما سبق

ليكن البيان التالي والذي يُمثل الطرق الممكنة بين مجموعة من المدن والمسافات بينها. وليكن المطلوب الانتقال من المدينة A إلى المدينة E:



9. يعطي تطبيق خوارزمية البحث العرض أولاً *Breadth-First* الحل:

a. ***ABE.***

b. *ABCE.*

c. *ACE.*

d. ولا خيار من الخيارات الثلاث السابقة.

10. يعطي تطبيق خوارزمية *A** مع تجريبية *h=0* الحل:

a. *ABE.*

b. *ACE.*

c. *ADE.*

d. **ولا خيار من الخيارات الثلاث السابقة.**

ليكن المطلوب الآن مسألة البائع الجوال على نفس البيان السابق والذي ينطلق من *A* ويعود إلى *A*:

11. يعطي تطبيق خوارزمية A^* مع تجريبية عدد الأقواس الناقصة * كلفة أكبر قوس الحل:

.ADBECA .a

.ADEBCA .b

<u>.ADCEBA</u> **.c**

.ADECBA .d

12. يعطي تطبيق خوارزمية A^* مع تجريبية عدد الأقواس الناقصة * كلفة أصغر قوس الحل:

<u>.ADBECA</u> **.a**

.ADEBCA .b

.ADCEBA .c

.ADECBA .d

13. تكون كلفة الحل الأمثل للمسألة:

.34 .a

<u>.36</u> **.b**

.38 .c

.40 .d

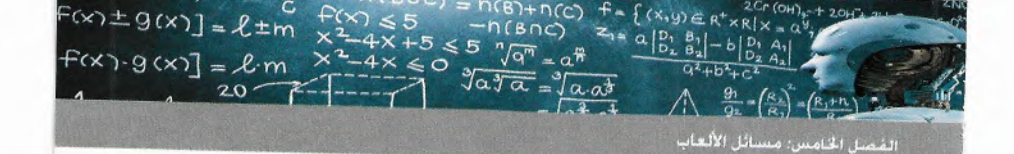
14. يوجد للمسالة بنفس الكلفة:

a. حل وحيد أمثل.

b. <u>حلين أمثلين.</u>

c. ثلاثة حلول أمثليه.

d. أربعة حلول أمثليه.



الفصل الخامس

مسائل الألعاب *Games*

نعالج الألعاب الحتمية مع لاعبين والتي تتميز:

- يتناوب اللاعبان باللعب، فيلعب اللاعب الأول، ثم الآخر، وهكذا دواليك.
- يعلم كل لاعب ماذا لعب اللاعب الآخر وما يُمكنه أن يلعب.
- تنتهي اللعبة بربح أحد اللاعبين (وخسارة اللاعب الآخر) أو التعادل.

مثال: لدينا بدايةً كومة مؤلفة من سبع ليرات، والهدف تقسيم هذه الكومة إلى عدة أكوام؛ على كل لاعب أن يقسم أحد الأكوام إلى كومتين غير متساويتين حصراً؛ الخاسر هو من لا يستطيع أن يلعب.

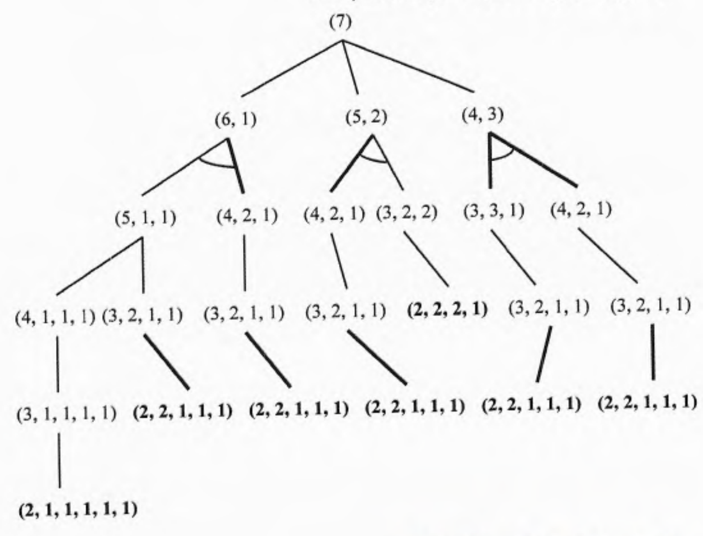
1 - 5 - شجرة اللعب

ليكن اللاعبان *J1* و *J2*، وليكن *J1* هو البادئ في اللعب. **تفيد قواعد اللعبة بإنشاء شجرة اللعب على النحو التالي:**

- يُمثّل الجذر (ذو المستوى *0*) موضع البداية.
- تُمثّل العقد ذات المستوى الزوجي المواضع التي على اللاعب *J1* أن يلعب فيها
- تُمثّل العقد ذات المستوى الفردي المواضع التي على اللاعب *J2* أن يلعب فيها

- تُمثّل الأقواس الصادرة من عقدة ما مختلف الضربات الممكنة والتي يمكن لعبها انطلاقاً من الموضع الممثل بهذه العقدة. وذلك من قبل اللاعب المعني (*J1* أو *J2* حسب شفعية المستوى للعقدة زوجياً أو فردياً).
- تُمثّل الأوراق المواضع الفائزة. أو الخاسرة. أو التي بلا مخرج.

تكون شجرة اللعب للعبة السابقة:



5 - 2 - خوارزمية *MinMax*

نسمّي اللاعبين من الآن فصاعداً *MAX* و *MIN*. وستكون مهمتنا إيجاد "أفضل" حركة للاعب *MAX* (الصديق). لنفترض أن اللاعب *MAX* سيلعب أولاً. ومن ثم يلعب اللاعبان بالتناوب.

تتألف الخوارزمية من إجرائيتين تستدعي كل منهما الأخرى (عودية متصالبة):

- الإجرائية *Maxmin* والتي تُستدعى من أجل عقد الصديق.
- الإجرائية *MinMax* والتي تُستدعى من أجل عقد الخصم.

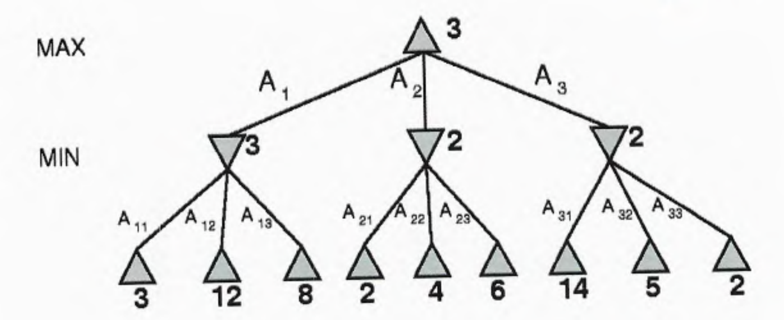
```
α ← Maxmin (R)
If  Leaf (R) Then
  α ← Evaluate (R)
Else
  α ← Max (Minmax (Succ_1 (R)),
     Minmax (Succ_2 (R)), ... Minmax (Succ_n (R)))
EndIf
```

```
β ← Minmax (R)
If  Leaf (R) Then
     β ← Evaluate (R)
Else
     β ← Min (Maxmin (Succ_1 (R)),
     Maxmin (Succ_2 (R)), ... Maxmin (Succ_n (R)))
EndIf
```

حيث:

- *Leaf(R)*: تُعيد *True* إذا كانت العقدة R ورقة.
- $Succ_i(R)$: تُعيد العقدة i الخَلَف لـ R.

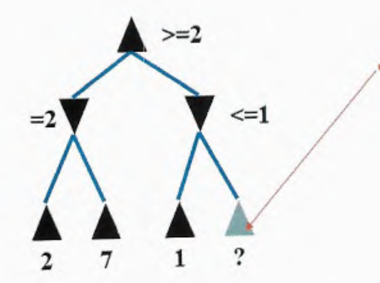
مثال:



5 - 3 - خوارزمية الفا-بيتا

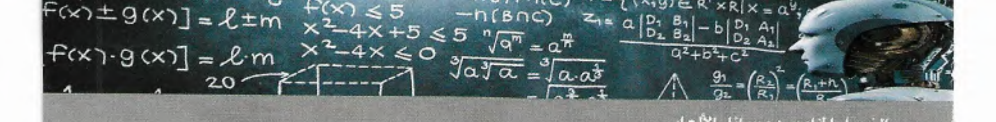
يُمكن تسريع وبشكل كبير أداء الخوارزمية *MinMax* السابقة باستخدام التشذيب α-β.

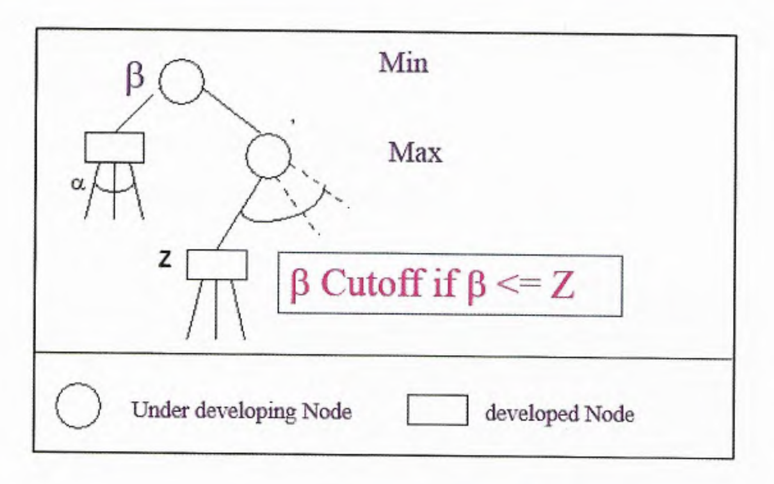
مثال:



5 - 4 - القطع بيتا *Beta Cutoff*

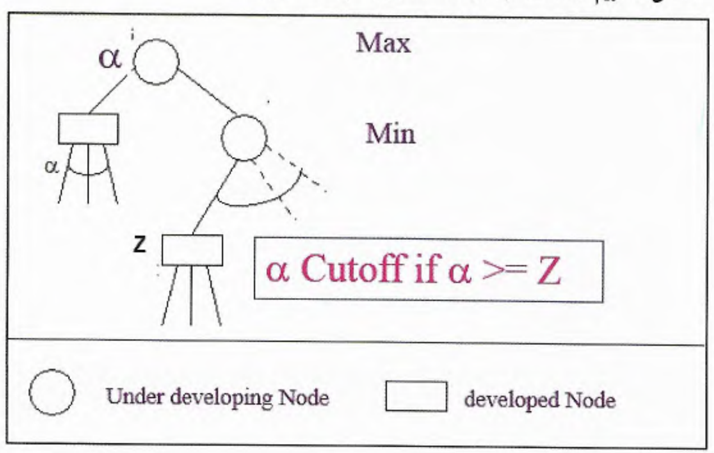
عندما نكون في عقدة *Max* تحت عقدة *Min* (تمّ حساب بيتا لها) فيُمكن التوقف عن تقويم العقد المتبقية للعقدة *Max* بمجرد الحصول على تقويم Z لأحد أبناء العقدة وبحيث: $\beta \leq Z$





5 - 5 - القطع ألفا *Alpha Cutoff*

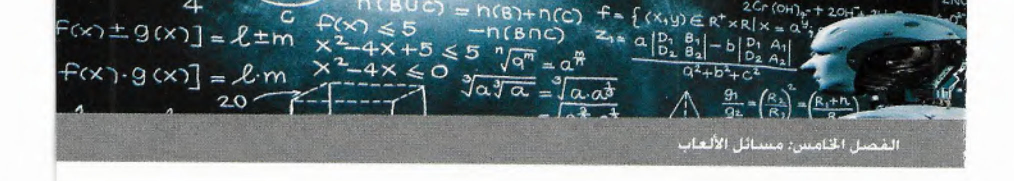
عندما نكون في عقدة *Min* تحت عقدة *Max* (تمّ حساب ألف لها) فيُمكن التوقف عن تقويم العقد المتبقية للعقدة *Min* بمجرد الحصول على تقويم Z لأحد أبناء العقدة وبحيث: $\alpha \geq Z$



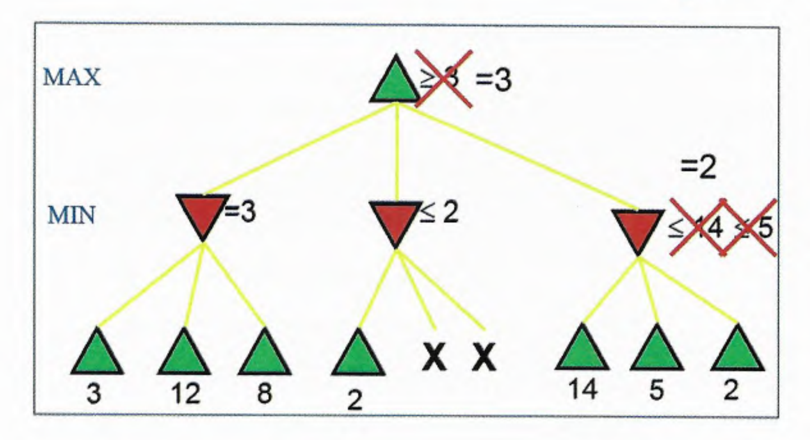
تتألف الخوارزمية من تابعين يستدعي كل منهما الآخر (عودية متصالبة):

- تابع يُستدعى عند عقد الصديق *Max*
- تابع يُستدعى عند عقد الخصم *Min*
- في البداية ($\alpha=-\infty$, $\beta=\infty$).

```
function MAX-value (n, alpha, beta)
if n is a leaf node then return f(n);
for each child m of n do
alpha :=max{alpha, MIN-value(m, alpha, beta)};
if alpha >= beta then return beta /* pruning */
end{do}
return alpha
function MIN-value (n, alpha, beta)
if n is a leaf node then return f(n);
for each child m of n do
beta :=min{beta, MAX-value(m, alpha, beta)};
if beta <= alpha then return alpha /* pruning */
end{do}
return beta
```



مثال



5 - 6 - أسئلة متعددة الخيارات

1. ليكن لدينا اللعبة التالية بثلاثة لاعبين: يوجد ثلاثة أبراج يحوي البرج الأول قرص والثاني قرص والثالث قرصين. يُمكن للاعب أن يأخذ قرص أو أكثر إنما من نفس البرج. اللاعب الذي يربح هو اللاعب الذي بعد أن يلعب لا يبقى أي قرص في الأبراج أي يجعل اللعبة *(0,0,0)*. تكون إذاً الحالة الابتدائية هي *(1,1,2)* ومنها يُمكن للاعب الأول أن ينقل اللعبة إلى أحد الحالات الثلاث التالية:

(2 ,1 ,0) يأخذ قرص من برج يحوي قرص

(0 ,1 ,1) يأخذ قرصين من البرج الذي يحوي قرصين

(1 ,1 ,1) يأخذ قرص من البرج الذي يحوي قرصين

ماذا يجب أن يلعب اللاعب الذي يبدأ كي يربح في النهاية:

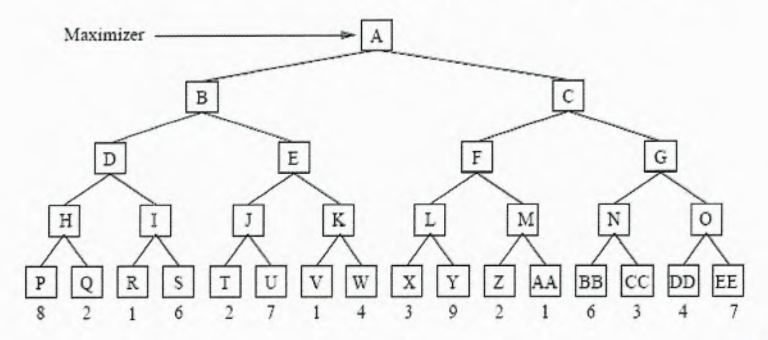
a. يأخذ قرص من برج يحوي قرص

b. **يأخذ قرص من البرج الذي يحوي قرصين**

c. يأخذ قرصين من البرج الذي يحوي قرصين

d. و لا خيار مما سبق

2. لتكن شجرة اللعب التالية وحيث أن اللاعب *MAX* هو الذي يبدأ:



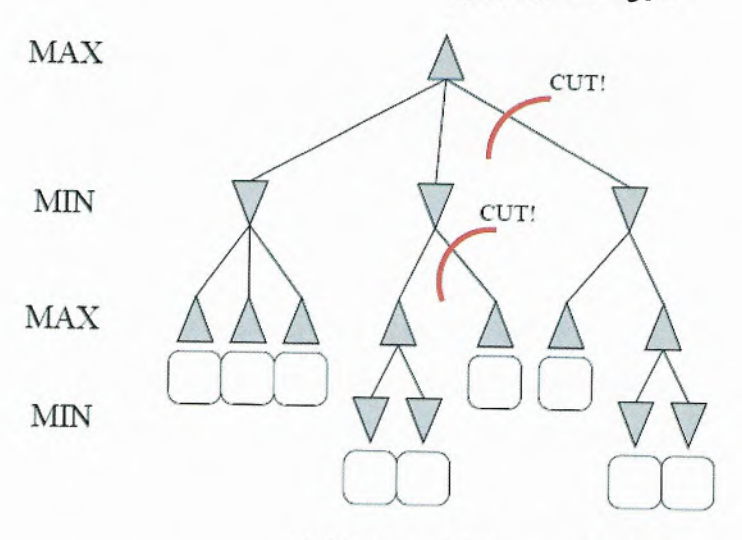
تقوم الخوارزمية ألفا-بيتا بحساب قيم الأوراق التالية

a. *P, Q, R, T, U, V, X, Y, Z, BB, CC, DD*

b. <u>***P, Q, R, T, U, X, Y, Z, BB, CC***</u>

c. *P, Q, R, T, U, V, X, Y, Z, BB*

3. لتكن شجرة اللعب التالية والمطلوب إيجاد قيم للأوراق كي نستطيع تحقيق القطعين المبينين



a. نضع قيم متساوية في جميع الأوراق.

b. نضع قيم بترتيب تنازلي لعقد *Max* و بترتيب تصاعدي لعقد *Min*.

c. نضع قيم بترتيب تنازلي لعقد *Min* و بترتيب تصاعدي لعقد *Max*.

d. <u>**ولا خيار من الخيارات الثلاث السابقة.**</u>

4. حدّد العبارة الصحيحة

a. لا تختلف قيم عقد *Min* و *Max* في خوارزميتي *Min-Max* و *Alpha-Beta*

b. لا يؤثر ترتيب الأوراق على فعالية خوارزمية *Alpha-Beta*

c. لا تتأثر فعالية الخوارزمية *Alpha-Beta* بمستوى الخصم

***d*. ولاخيار مما سبق**

5. ليكن لدينا اللعبة التالية: يوجد كومة من *n* قطعة. يقوم اللاعب حين يأتي دوره بأخذ على الأقل قطعة واحدة وعلى الأكثر *m* قطعة من الكومة. يربح اللاعب الذي يأخذ آخر قطعة من الكومة. حدّد الجملة الصحيحة فيما يلي

a. يربح اللاعب الذي يبدأ باللعب دائما

b. يخسر اللاعب الذي يبدأ باللعب دائما

c. يربح اللاعب الذي يبدأ إذا كانت *n* من مضاعفات *m*

d*. يربح اللاعب الذي يبدأ إذا كانت *n* ليست من مضاعفات *m+1

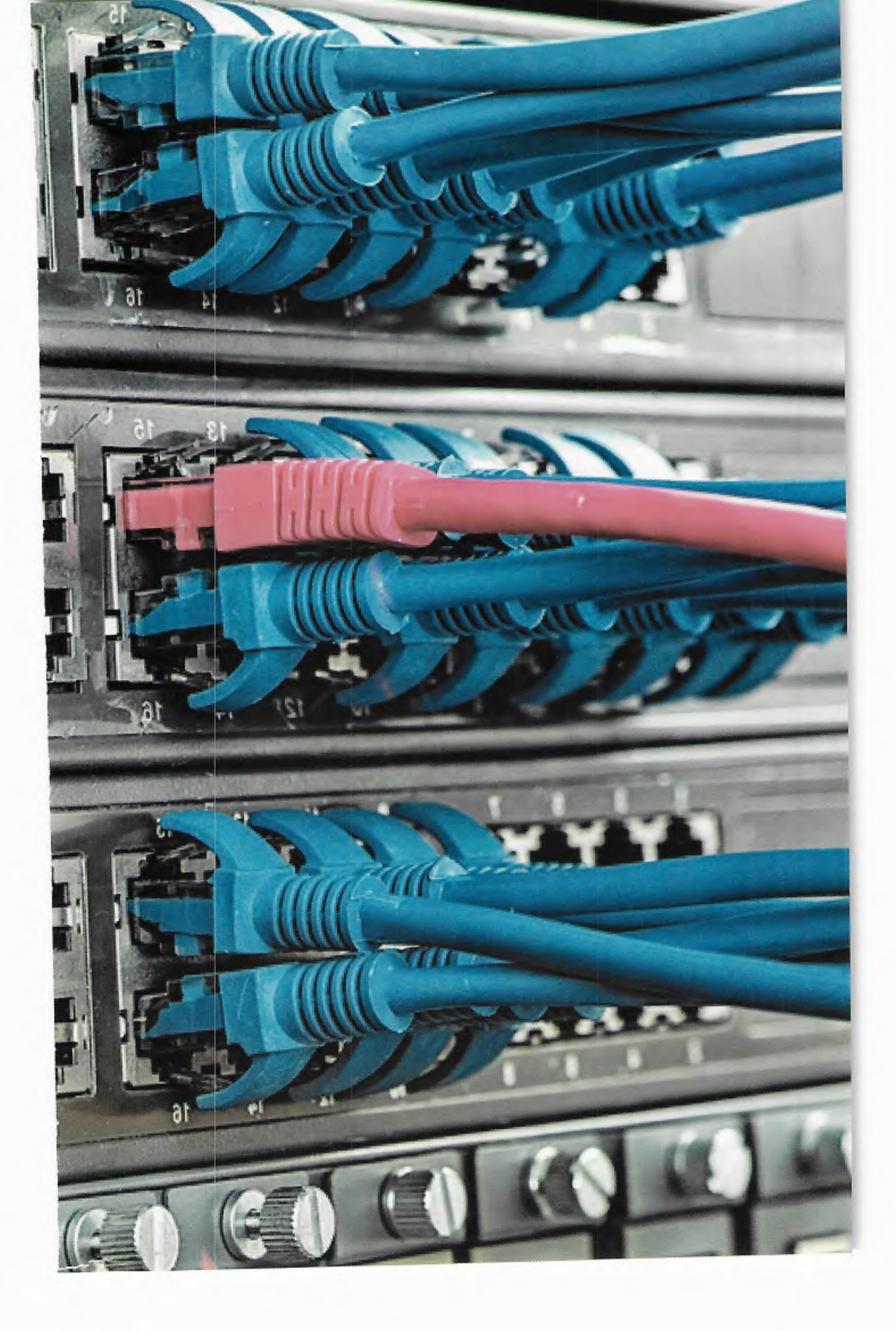
مثال لفهم اللعبة: إذا كانت مثلا *n=20* و *m=3* فإن اللاعب الذي يبدأ يمكن أن يأخذ قطعة أو اثنتين أو ثلاثة.

يربح اللاعب الذي حين يأتي دوره يكون في الكومة قطعة (فيأخذها) أو قطعتين (فيأخذهما) أو ثلاثة قطع (فيأخذهم).

الباب الرابع

الشبكات

(Networks)





الفصل الأول

مدخل إلى الشبكات الحاسوبية

1 - 1 - تعريف الشبكة

الشبكة الحاسوبية هي مجموعة من الحواسيب المرتبطة مع بعضها البعض بطريقة ربط معينة وعبر وسائط نقل بيانات مختلفة *(سلكية أو لاسلكية)* عادة ما تتبع لمعايير خاصة متنوعة.

تهدف الشبكة بشكل رئيسي تبادل المعلومات والبيانات المتاحة، والموارد مثل الطابعة، والخدمات *(Services)* مثل قواعد البيانات أو البريد الالكتروني، فيما بينها على الشبكة أو البرامج التطبيقية أياً كان نوعها *(مثل محرر نصوص)*، وكذلك تسمح بالتواصل المباشر بين المستخدمين.

تتكون الشبكة في ابسط حالاتها من جهازين متصلين مع بعضها البعض بواسطة سلك ويقومان بتبادل البيانات فيما بينهما. ويسمى الوسط الذي يستخدم لنقل البيانات أو الخدمات بين الحواسيب بوسط الإرسال *(Transmission Medium)*.

ويمكن للشبكات الحاسوبية أن توصل مع بعضها البعض لتشكل شبكات أوسع؛ وتعتبر الانترنت مثالاً معروفاً عن شبكة الشبكات.

تهدف الشبكات الحاسوبية إلى:

1 - تخفيض كلفة التجهيزات عبر مشاركة الموارد والبرمجيات غالية الثمن وتحقيق معدل استخدام عال لها.

2- نقل البيانات والمعلومات بين المستخدمين بطريقة مباشرة وآنية.

3- توفير إمكانية الإدارة المركزية لهذه الحواسيب من أماكن مختلفة وبعيدة وكذلك إدارة المستخدمين والموارد لهذه الشبكة.

1 - 2 - الاتصال بين الإجراءات *Process Communications*

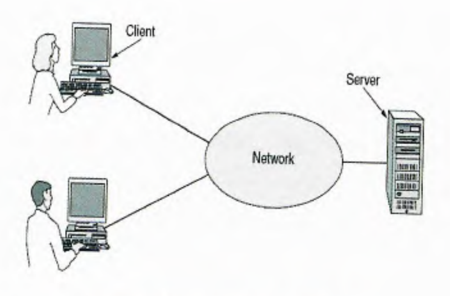
يتألف البرنامج عادةً من مجموعة من الإجراءات التي تتخاطب مع بعضها. تتخاطب الإجراءات ضمن المضيف نفسه باستخدام الاتصال بين الإجراءات الذي يتيحه نظام التشغيل. ما يهمنا هنا، هو طريقة التخاطب بين إجراءات تابعة لأجهزة مختلفة.

يجري التخاطب بين إجراءات تابعة لأنظمة مختلفة عن طريق تبادل الرسائل عبر الشبكات الحاسوبية. تسمى الحواسيب التي تقدم البيانات أو الموارد أو خدمات أخرى كخدمة الويب في الشبكات الحالية باسم مخدمات الشبكة *(Servers)*، بينما تسمى الحواسيب التي تطلب وتستفيد من هذه البيانات أو الخدمات باسم الزبائن *(Clients)*. ويمكن لجهاز واحد أن يقوم بدور المخدم والزبون في نفس الوقت.

هناك عدة أنواع من المخدمات بحسب عملها بشكل عام، ومنها:

1. مخدمات الملفات *(File Servers)*.
2. مخدمات الطباعة *(Print Servers)*.
3. مخدمات التطبيقات *(Application Servers)*.
4. مخدمات قواعد البيانات *(Database Servers)*.





الشكل *1-1*: النموذج المخدم-الزبون.

ويسمى هذا النموذج من بنى الشبكات بنموذج المخدم-الزبون وهو نموذج كثير الانتشار ويشكل أساس استخدام معظم الشبكات الحاسوبية.

من جهة أخرى يعتبر نموذج الند-للند *(Peer-to-Peer)* أيضاً من نماذج الشبكات الشائعة حيث يشكّل عدد من الأجهزة المستقلة مجموعة تستطيع الاتصال فيما بينها ويستطيع كل جهاز في المجموعة تأدية وظائف الزبون والمخدم في نفس الوقت. ويكون لكافة الأجهزة ذات الأهمية والأولوية والحقوق ولا تحوي مثل هذه الشبكات على مخدم مخصص وتنتمي إلى شبكات الإدارة الموزعة.

الكثير من أنظمة الند-للند مثل *(BitTorrent)* لا تحتوي على قاعدة بيانات مركزية للمحتويات. وعلى العكس فان كل مستخدم يحتفظ بقاعدة بياناته محلياً ويزود الشبكة بلائحة عن المستخدمين المجاورين له والأعضاء في الشبكة. غالباً ما تستخدم اتصالات الند-للند لتبادل الفيديو والموسيقى، والتي عادة هذه الأنظمة تتضمن تفاعلاً بين مستخدم وقاعدة بيانات بعيدة. النوع الآخر من استخدام هذا النموذج من الشبكات يضمن اتصال مستخدم-إلى-مستخدم مثل الرسائل الآنية *(Instant Messaging)*، وكذلك يوجد خدمات رسائل



متعددة المستخدمين مثل خدمة *(Twitter)* التي ترسل رسالة إلى كافة الأصدقاء في المجموعة. نوع آخر من هذه التطبيقات هو تطبيقات الشبكات الاجتماعية *(Social Networks)*. كذلك يمكن لمجموعة من الأشخاص العمل معاً لتوليد محتوى على الشبكة باستخدام هذا النموذج من الشبكات مثل *(Wiki)* الذي هو صفحة ويب تعاونية والتي يقوم أعضاء المجموعة بتنقيح محتوياتها ومن أشهرها هو موسوعة *(Wikipedia)*.

تحتاج الشبكات إلى برنامج شبكي مثبت على كافة تجهيزات الشبكة وعادة ما يكون هذا البرنامج هو نظام تشغيل شبكي *(Network Operating System –NOS)*

1 - 3 - تصنيف الشبكات

يمكن تصنيف الشبكات وفق عدة معايير، والمعايير الأكثر استخداماً هي:

1. حسب الانتشار الجغرافي.
2. حسب الطبوغرافية الفيزيائية للشبكة.

1 - 3 - 1 - تصنيف الشبكات حسب الانتشار الجغرافي

يمكن للشبكات وفق هذا التصنيف أن تقسم إلى الأنواع التالية:

1. الشبكة الشخصية *(Personal Area Network)*.
2. الشبكة المحلية *(Local Area Network-LAN)*.
3. شبكة المدن أو الشبكات الإقليمية *(Metropolitan Area Network-MAN)*.
4. الشبكة الواسعة *(Wide Area Network-WAN)*.

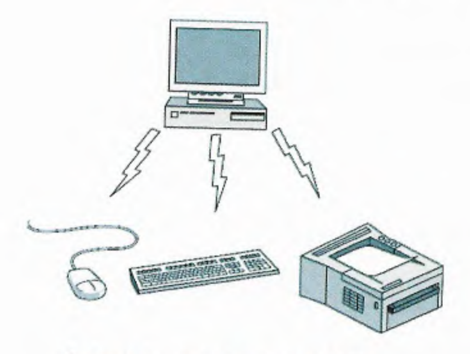


1 - الشبكة الشخصية *(PAN)*

الشبكات الشخصية هي شبكات حواسيب المستخدمة للتواصل بين أجهزة الحاسب القريبة من بعضها البعض ومن المستخدم وتتضمن هذه الشبكات أجهزة الهاتف والمساعدات الرقمية. تسمح هذه الشبكات للأجهزة المعرّفة ضمنها بالاتصال على نطاق المستخدم ومدى هذا النوع من الشبكات عدة أمتار فقط.

يمكن استخدامها للتواصل بين الأجهزة بعضها البعض، أو للاتصال بمستوى أعلى من الشبكات. ويوصل النوع السلكي من هذه الشبكات عن طريق *(USB)* أو الفايروابر *(Firewire)*.

الأكثر شيوعاً من هذا النوع من الشبكات هو شبكات لاسلكية قصيرة المدى؛ تربط الحاسب مع تجهيزاته الطرفية مثل الفارة، ولوحة المفاتيح، والطابعة باستخدام البلوتوث *(Bluetooth)*. وفي أبسط أشكالها تستخدم نموذج السيد-العبد *(Master-Slave)* في الاتصال حيث يعمل الحاسب كالسيد والأجهزة الطرفية تعمل كالعبيد بالاستجابة للطلبات المقدمة من السيد.



الشكل *1-2*: شبكة شخصية لاسلكية

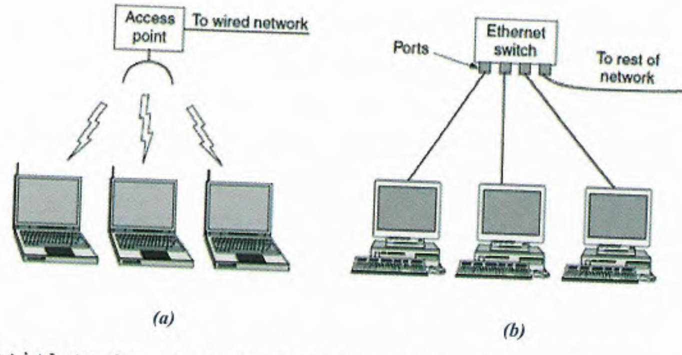
2 - الشبكة المحلية *(LAN)*

الشبكات المحلية هي شبكات تستخدم لتغطية أماكن محدودة وصغيرة مثل المنزل أو المكتب أو مؤسسة صغيرة وهي عادة ما تكون شبكة خاصة تعمل ضمن نطاق بناء واحد أو مجموعة أبنية متجاورة.

أصبحت الشبكات المحلية اللاسلكية *(Wireless LAN)* شائعة بكثرة في هذه الأيام؛ بالأخص في المنازل والمكاتب. في معظم الحالات يتصل كل حاسب مع جهاز طرفي عادة ما يوضع في منتصف المنزل أو المكتب ويسمى بنقطة النفاذ *(Access Point – AP)*، أو الموجه اللاسلكي *(Wireless Router)*، أو المحطة القاعدية *(Base Station)*. مهمة هذه الطرفية نقل الرزم *(Packets)* بين الحواسيب اللاسلكية وأيضاً بينهم وبين الانترنت. المعيار الأشهر للشبكات المحلية اللاسلكية هو المعيار *IEEE 802.11* والمسمى بـ *(WiFi)*. يمكن لسرعات النقل ضمن هذا المعيار أن تتراوح بين *11* إلى مئات الميغا بيت في الثانية *(Mbps)*.

هناك عدة طرق لوصل الشبكات المحلية السلكية مثل الكابلات النحاسية أو الكابلات الضوئية. عادة ما تستخدم الشبكات المحلية السلكية من الوصلات المباشرة *(Point-to-Point)*. المعيار *IEEE 802.3* والمسمى بالإيثرنت هو المعيار الأكثر شهرة للشبكات المحلية السلكية. بالإضافة يستخدم عدد من التقنيات الشبكية مع معايير تابعة لها لتشكيل الشبكات المحلية من أشهرها:

- شبكة وفق المعيار *(Token Ring)*.
- شبكة وفق المعيار *(Fiber Distributed Data Interface – FDDI)*.
- شبكة وفق المعيار *(Gigabit Ethernet)*.
- شبكة وفق معيار *(Asynchronous Transfer Mode-ATM)*.



الشكل 1-3: *(a)* شبكة محلية لاسلكية *(802.11)* - *(b)* شبكة محلية سلكية إيثرنت

وسيتم شرح بعض من هذه المعايير في الفصل الرابع.

من المحتمل أيضاً تقسيم شبكة محلية فيزيائية إلى شبكات محلية أصغر منطقية مما يسهّل إدارتها وتسمى بالشبكات المحلية الافتراضية *(Virtual LAN-VLAN)*.

يساعد قياس الشبكة المحلية *(التغطية الجغرافية)* المحدود في إعطاء حرية أكبر في اختيار وسائط النقل كما يسهل عملية الإدارة. أما بالنسبة لوسائط النقل فيمكننا هنا تمديد كبل واحد يربط جميع المحطات ويتميز بمعدل نقل معطيات عالٍ، ومعدل تأخير قليل، ومعدل أخطاء منخفض.

يمكن توصيل الشبكات المحلية مع بعضها البعض عن طريق موصلات من الشبكات الواسعة وذلك باستخدام الموجهات *(Routers)*.

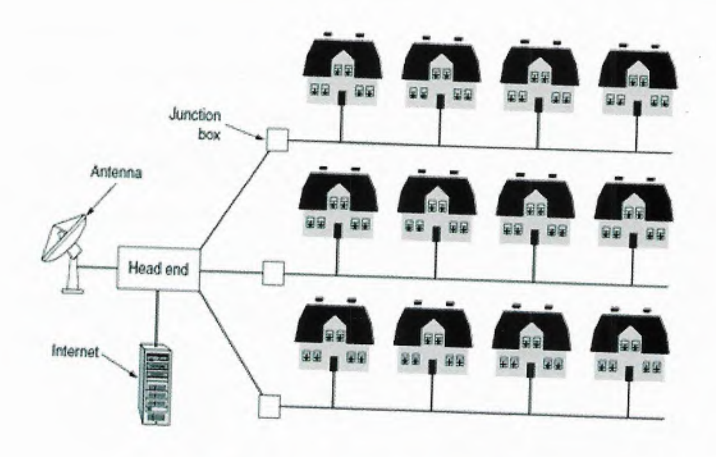
يتراوح عدد أجهزة الحاسب في هذا النوع من الشبكات بين جهازين على الأقل إلى *500* جهاز وتستخدم المجمعات *(Hubs)* والمبدلات *(Switches)* لربط الأجهزة مع بعضها البعض وتمكينها من الاتصال فيما بينها.

تعتمد الشبكات المحلية على التعميم *(البث الإذاعي)* *(Broadcasting)* كأساس لنقل المعطيات؛ أي أنه عندما تضع محطة ما الإطار المراد إرساله على وسيط النقل فإن بقية المحطات تستلم نسخة عن الإطار وتقوم بتوصيله إلى الطبقة الأعلى في حال كان عنوان الوجهة الفيزيائي يطابق عنوانها أو إذا كان الإطار معمماً في الأصل على الجميع. وتقوم بإهماله في بقية الحالات.

3 - شبكات المدن *(MAN)*

وهي الشبكات التي تغطي المدن الكبرى وتتكون من مجموعة من الشبكات المحلية في المدينة وقد يصل مداها إلى 50 كم. صمّم هذا النوع من الشبكات لنقل البيانات عبر مناطق جغرافية شاسعة وهي تصلح لربط مدينتين متجاورتين مع بعضهما.

تعتبر شبكات نقل التلفزيون عبر الكبل من أكثر شبكات المدن شيوعاً؛ فهي تسمح بتغطية مدينة كاملة. بدأت هذه الشبكات، كما يدل اسمها، ببث بعض محطات التلفزيون إلى المنازل، ثم سرعان ما بدأت تحسن الخدمات التي تزودها حتى وصلت إلى تزويد خدمة الإنترنت بأسعار رمزية. يعود ذلك لكون الكبلات المحورية المستخدمة ممددة مسبقاً ولكون تعديل شبكة البث التلفزيوني لنقل الإشارات الرقمية التي يولدها الحاسب غير مكلفة. يوجد حالياً أنواعاً أخرى من شبكات المدن اللاسلكية مثل شبكات المعيار *(IEEE 802.16)* المسمى *(WiMAX)* والتي تسمح بتغطية مدينة كاملة وبمعدل نقل معطيات يصل إلى *70 Mbps*. وهناك أيضاً شبكة *(Metro Ethernet)* التي يزداد استخدامها لتغطية مدينة كاملة.



الشكل 1-4: شبكة المدن المعتمدة على كابل التلفاز

4 - الشبكات الواسعة (WAN)

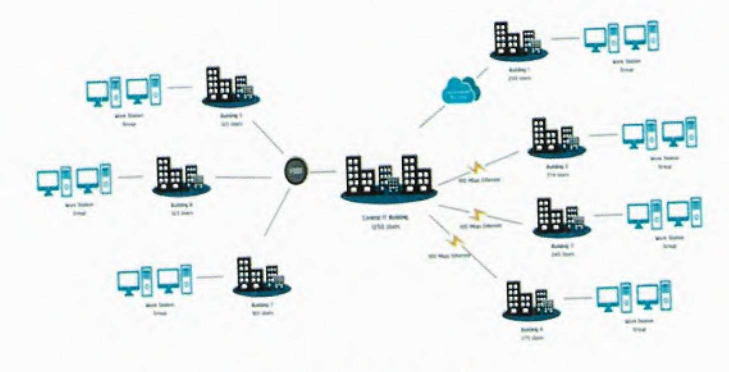
هي مجموعة من الشبكات الصغيرة المتصلة مع بعضها البعض ويمكن أن تمتد إلى عدة دول أو عدة قارات مترامية الأطراف. وتستخدم هذه الخاصية بشكل واسع من قبل الشركات المصنعة لأجهزة الهاتف النقال لربطها بالشبكة العالمية الانترنت بصورة سريعة باستخدام تقانات مختلفة مثل خدمة *(General Packet Radio Service –GPRS)*. وتعمم هذه الخاصية في المطارات الضخمة والأماكن الهامة والمزارات السياحية وليس لها أي تكلفة على مستخدميها بل هي خدمة مجانية مقدمة للمستهلك. غالباً ما تكون الحواسب في الشبكات الواسعة موصولة إلى الشبكة العمومية، ويمكن أيضا أن توصل من خلال خطوط مؤجرة من شركات الاتصالات.

تحوي الشبكات الواسعة على محطات عمل، يطلق عليها اسم المضيف *(Host)*، يستخدمها المستثمر لتشغيل برامج تطبيقية (أو تطبيقات). يجري ربط المضيفين باستخدام شبكة اتصال جزئية



(Communication subnet). تعود ملكية المضيف للمستثمر بينما عادة ما تعود ملكية الشبكة الجزئية إلى أحد مؤسسات الاتصالات أو مزود خدمة الإنترنت *(Internet Service Provider –ISP)* الذي يكون مسؤولاً عن تشغيلها.

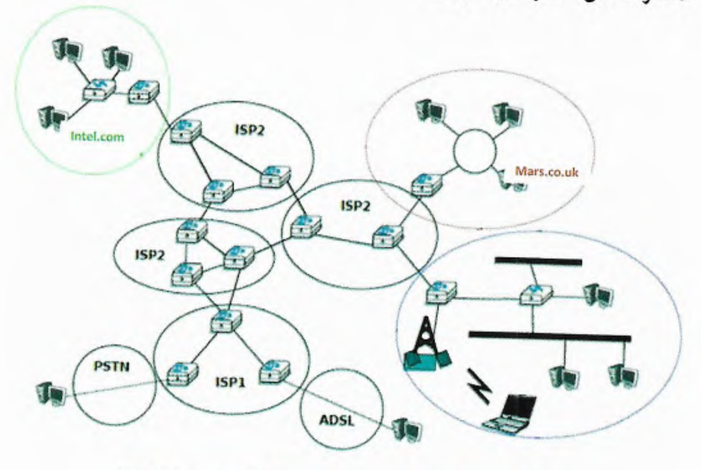
تتمثل وظيفة الشبكة الجزئية بنقل الرسائل من مضيف إلى مضيف آخر بطريقة تسمح بفصل تفاصيل تطبيقات الشبكة عن تفاصيل النقل الفيزيائية البحتة الأمر الذي يفيد في تبسيط مهمة تصميم الشبكات.



الشكل *1-5*: الشبكة الواسعة

تتألف الشبكة الجزئية من خطوط الاتصال *(Transmission Lines)*، وعناصر تبديل *(Switching Elements)*، أو موجهات *(Routers)*. تنقل خطوط الاتصال، مثل الكبلات النحاسية، أو الضوئية، أو الوصلات الميكروية اللاسلكية، البيانات من محطة إلى أخرى؛ بينما تقوم عناصر التبديل بربط ثلاثة خطوط اتصال أو أكثر مع بعضها البعض. وعندما تدخل المعطيات على أحد خطوط الاتصال يعمل عنصر التبديل على توجيهه إلى خط اتصال الخرج المناسب.

تقوم الشبكات الواسعة بربط شبكات محلية تعمل على تقانات اتصال مختلفة. فيمكن لجزء منها أن يكون شبكة إيثرنت محلية تتصل مع الأجزاء الأخرى من الشبكة عبر خطوط اتصال بعيدة المدى مثل خطوط *(SONET)*. وبذلك تحتاج هذه الشبكات الواسعة إلى أجهزة قادرة على ربط الأنواع المختلفة من أوساط النقل عبر التشبيك البيني *(Interworking)* الذي يعمل على ترابط الشبكات المركّبة المكونة من أكثر من شبكة.



الشكل *1-6* : شبكة بينية بسيطة *(An Internetwork)*

أخيراً يمكن لشركة بدلاً من تأجير خطوط نقل مخصصة من قبل شركات الاتصالات لربط مكاتبها المتناثرة والبعيدة، أن تقوم بربط هذه المواقع المتباعدة عبر الانترنت مما يسمح لهذه الاتصالات بين المكاتب أن تكون عبر خطوط اتصال افتراضية تستخدم البنية التحتية المتوفرة في الإنترنت. يسمى هذا الترتيب بالشبكة الخاصة الافتراضية *(Virtual private Network –VPN)*. يعتبر هذا النمط أرخص من تأجير خطوط اتصال مخصصة للشركة ولكنه من جهة أخرى يعاني من

نقص التحكم بالموارد في البنية التحتية التابعة لشبكة الانترنت. ففي الخطوط المخصصة تكون سعة الخط واضحة ومعلومة. أما في الشبكات الافتراضية الخاصة *(VPN)* فيمكن لهذه السعة أن تتغير وفقاً لخدمة الانترنت.

يمكن للشبكات الواسعة أن تستخدم الكثير من تقانات اللاسلكية مثل أنظمة الأقمار الصناعية *(Satellite Systems)*. وكذلك تعتبر شبكات الهاتف الخلوي مثالاً عن الشبكات الواسعة التي تستخدم التقنيات اللاسلكية.

يبين الجدول التالي مقارنة بين الفروق الرئيسية لتصنيف الشبكات حسب الانتشار الجغرافي:

المسافة بين الأجهزة	الأجهزة موجودة ضمن مجال	مثال
متر واحد	متر مربع	الشبكات الشخصية
10 أمتار	غرفة	الشبكات المحلية
100 متر	بناء	
1 كم	حرم	
10 كم	مدينة	الشبكات الإقليمية
100 كم	دولة	الشبكات الواسعة
1000 كم	قارة	
10000 كم	الكرة الأرضية	الإنترنت

الجدول *1-1*: تصنيف الشبكات حسب المسافة بين الأجهزة



1 - 3 - 2 - تصنيف الشبكات حسب الطبوغرافية / الطبولوجية الفيزيائية

ويقصد بطبولوجية الشبكة طريقة توصيل العقد في الشبكة هندسياً. ويقصد بالعقد أجهزة الحاسب أو الاتصالات المختلفة. وتعّف بعض خصائص الشبكة مثل مواقع توضع العقد والترتيب المحدد لكافة الوسائط الفيزيائية المستخدمة في التوصيل مثل الكابلات.

يؤثر اختيار أحد تشكيلات الشبكة الفيزيائية دون الآخر على كل من التالي:

1. أدوات إدارة الشبكة.
2. نوع المعدات وإمكانياتها التي تحتاجها الشبكة.
3. مقدار نمو الشبكة مستقبلاً.

ومن أكثر أنواع طبولوجية الشبكات المستخدمة هي:

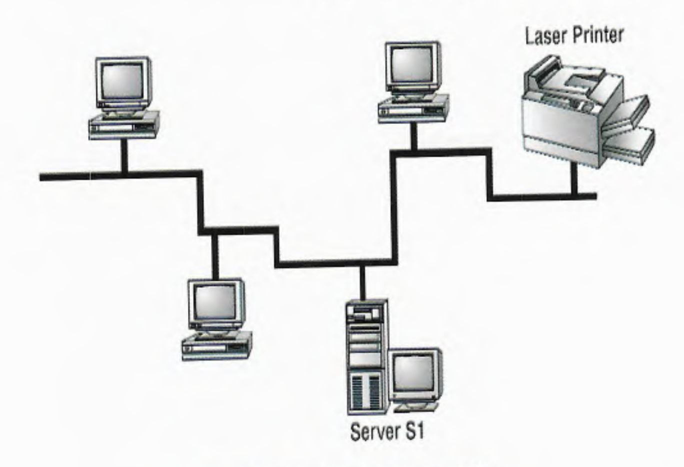
1. شبكات الناقل العمومي *(BUS)* ويسمى أيضاً بالعمود الفقري *(Backbone)*.
2. الشبكات النجمية *(Star)*.
3. الشبكات الحلقية *(Star)*.
4. الشبكات التشابكية *(Mesh)*.
5. الشبكات الشجرية *(Tree)*.



1 - شبكة الناقل العمومي *(Bus)*

يعتبر تصميم الشبكة من هذا النوع هو الأبسط والأسهل من حيث الوصل والتركيب، وربما الأكثر شيوعاً في الشبكات المحلية. يعتمد هذا التصميم على وصل الأجهزة الحاسوبية في صف وعلى طول كبل وحيد يدعى بالمقطع الشبكي *(Segment)*.

يستطيع أي جهاز على هذه الشبكة أن يرسل إلى أي عقدة أخرى وتنتقل الرسالة إلى كافة العقد الموجودة على الشبكة ولكن لا يستطيع قراءتها إلا العقدة المرسلة لها والمحددة بالعنوان الموجود في الرسالة. ويكون المرسل في هذه اللحظة هو المسيطر على الشبكة (مالكاً لوسط النقل حتى ينتهي من عملية الإرسال). ويمتلك هذا النوع من الشبكات نهايتين متميزتين.



الشكل *1-7*: طبولوجية الناقل العمومي

ويعتمد هذا النوع من تصاميم الشبكات على إرسال الإشارة *(Signal)*، ومن ثم ارتداد الإشارة *(Signal Bounce)*، وأخيراً من المنهي



أو موقف الإشارة *(Terminator)*. فعندما ترسل إشارة بيانات على هذا النوع من الشبكات فإنها تنتقل على كامل كبل التوصيل وصولاً إلى نهايته، وستبدأ بالارتداد جيئة وذهاباً. إذا لم يتم مقاطعة هذه الإشارة، مما يتسبب بمنع باقي الأجهزة في الشبكة من تملك وسط النقل وإرسال بياناتها.

لذلك يعمل على إيقاف الإشارة بعد وصولها إلى الجهاز المطلوب؛ ويتم ذلك عبر موقف الإشارة الذي يتم وضعه عند كل طرف من أطراف كبل الشبكة، ويوصل بكل حاسب متصل بالشبكة، مهمته امتصاص الإشارة من الكبلات وأسلاك التوصيل.

إذا ما قام جهازين على هذا النوع من الشبكات بالبدء بإرسال البيانات في نفس الوقت فسيحدث تصادم بين الرسائل *(Collision)*، ولهذا على كل حاسب انتظار دوره لإرسال البيانات على الشبكة، وبالتالي كلما زاد عدد الأجهزة على الشبكة كلما ازداد الوقت الذي على الجهاز انتظاره ليحل دوره في الإرسال، وبالتالي ازداد بطأ الشبكة.

تستخدم تقنية تعرف بالنفاذ المتعدد بتحسس الإشارة الحاملة مع كشف التصادم *(Carrier Sense Multiple Access with Collision Detection - CSMA / CD)* والمستخدمة في شبكات الإيثرنت التي تقوم على تحسس الكبل من قبل أي مرسل قبل بدء الإرسال فإذا وجده مشغولاً ينتظر حتى ينتهي، وإذا ما تم بدء الإرسال من جهازين في نفس اللحظة يتوقف كلاهما لمدة عشوائية من الزمن قبل إعادة المحاولة.

ايجابيات شبكة الناقل العمومي:

1. سهولة التركيب والتوسيع عبر إضافة أجهزة جديدة.
2. التكلفة البسيطة بسبب وجود كابل وحيد.

سلبيات شبكة الناقل العمومي:

1. محدودية عدد الأجهزة وعدد نقاط الربط لأن امتداد الكبل محدود.

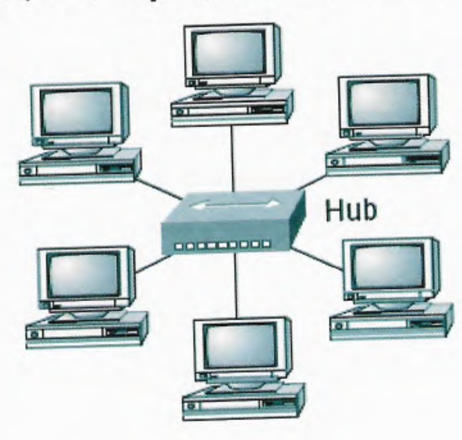
2. وجود خطأ أو انقطاع في الكبل الرئيسي أو في أي عقدة يسبب تعطل وتوقف الشبكة بالكامل.

3. تعتبر أبطأ من أنواع الشبكات الأخرى.

4. صعوبة تعقب واستكشاف الأخطاء، التعديل والنقل.

2 - الشبكة النجمية *(Star)*

في هذا النوع من الشبكات تربط أجهزة الحاسب عبر أسلاك موصلة وكابلات أو عبر الوصلات اللاسلكية بجهاز مركزي يسمى بالمجمع *(Hub)* أو بمبدلة *(Switch)* أو نقطة نفاذ *(Access Point)*. يمكن للمجمع أن يكون فعالاً (نشطاً) *(Active)* فيقوم بتقوية الإشارات المارة عبره، ويمكن توصيل عدد من المجمعات المركزية في الشبكة الواحدة.



الشكل *1-8*: الطبولوجية النجمية

يعتبر هذا النوع من وصل الشبكات أفضل الأنواع؛ وتقدم الكثير من الميزات على شبكة الناقل العمومي مما يجعلها النوع الأكثر استخداماً ولكنها تحتاج أيضاً إلى وسائط نقل فيزيائية أكثر. من أهم حسناتها، وبسبب كون كل جهاز أو مقطع شبكة يتصل بكبل بطريقة مستقلة إلى الجهاز المركزي في الشبكة، فإن أي عطل أو انقطاع لهذا الكبل يعطل الجهاز أو المقطع الشبكي المتواجد على هذا الكبل فقط. هذا ما يجعل استكشاف وتعقب الأخطاء أسهل في هذا النوع، ويجعل الشبكة مواربة للأخطاء *(Fault Tolerance)*. الميزة الأخرى لهذا النوع هو أنها أكثر قدرة على التوسع عبر ربط كبل جديد إلى الجهاز المركزي.

من أهم سيئاتها كونها مركزية؛ فيسبب فشل الجهاز المركزي فيها فشل الشبكة بالكامل، ولكن في العادة نادراً ما تفشل المجمعات والمبدلات والأجهزة المركزية الأخرى المستخدمة.

يمكن تلخيص حسنات الشبكات النجمية بـ:

1. القدرة العالية على التوسع عبر إضافة الأجهزة الجديدة بسهولة وسرعة.

2. لا يسبب فشل أحد الكابلات في الشبكة فشل كامل الشبكة.

3. سهلة لاستكشاف الأخطاء وهي مواربة للخطأ.

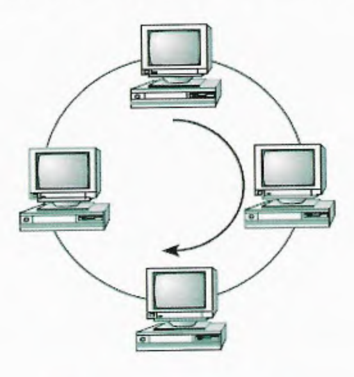
أما سيئات الشبكات النجمية هي:

1. يمكن لكلفة التركيب الكلية أن تكون عالية بسبب عدد الكابلات الكبير.

2. لديها نقطة فشل مركزية هي الجهاز المركزي.

3 - الشبكة الحلقية (Ring)

هي شبكة تكون على الشكل الدائري على الأقل من الناحية النظرية؛ حيث تنتقل الإشارات من عقدة إلى أخرى في اتجاه واحد فقط. وتتصل كل عقدة مع عقدتين بشكل مباشر؛ عقدة ترسل لها وعقدة تستقبل منها. تشارك العقد بشكل فعال بتمرير الرسائل عبر الشبكة وتقوم في بعض الحالات بتقوية الإشارة قبل تمريرها إلى العقدة التالية. ولكن بسبب كون الإشارة تمر على كافة الأجهزة في الشبكة فإن فشل أو توقف أي جهاز على الشبكة يسبب توقف كامل الشبكة عن العمل.



الشكل 1-9: الطبولوجية الحلقية

تعتبر الشبكة الحلقية كثيرة التشابه مع شبكة الناقل العمومي؛ فعند الرغبة بإضافة عقد جديدة للشبكة علينا إيقاف كامل الشبكة عن العمل بسبب كسر الكبل الحلقي. هذا من أهم أسباب كون هذا النوع من الشبكات غير شائع بالإضافة إلى كونه عالي الكلفة لاستخدام عدة كابلات لوصل كل جهاز إلى الشبكة. عادة ما يستخدم هذا النوع من الشبكات في الشبكات الواسعة *(WAN)* مثل *(SONET)* ولا تستخدم في الشبكات المحلية.

يستخدم هذا النوع من الشبكات تقنية تمرير الإشارة *(العلامة) (Token Passing)*؛ فعندما يريد جهاز ما إرسال البيانات عليه أن ينتظر دوره حتى يستلم علامة حرّة *(Free Token)* تعلمه بقدرته على البدء بإرسال بياناته عبر الشبكة، يضيف الجهاز الراغب بالإرسال بياناته على العلامة الحرة، ويحدد عنوان جهاز المستقبل ثم يرسل هذه العلامة.

تبدأ العلامة بالانتقال بين جهاز إلى آخر حتى تصل إلى الجهاز الذي يتطابق عنوانه مع العنوان المضاف إلى الرسالة.

حسنات الشبكة الحلقية

1. قصر كبلها الرئيسي مما يساعد باستخدام الألياف الضوئية عالية الكلفة.

2. قابلية التوسع في الشبكة.

أما سيئات الشبكة الحلقية، كما شبكة الناقل العمومي:

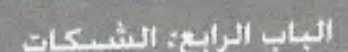
1. إيقاف عمل كامل الشبكة عند التوسيع.

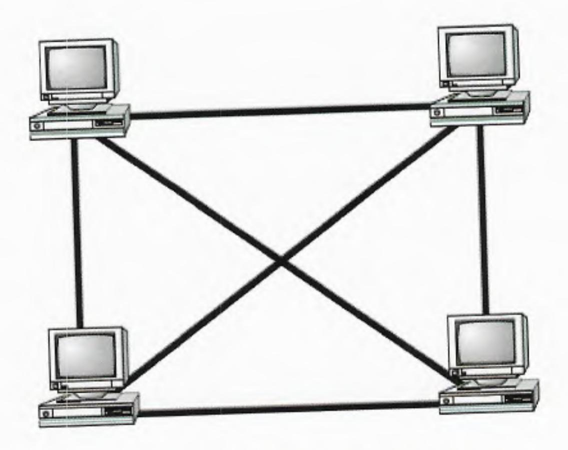
2. تعطل الكبل الرئيسي يسبب تعطل وإيقاف كامل الشبكة.

يمكن تشكيل حلقة مضاعفة في هذا النوع من الشبكات لمضاعفة وزيادة وثوقية الشبكة.

4 - الشبكة التشابكية (Mesh)

هذا النوع من الشبكات قليل الاستعمال، بل نادرًا ما يتم إنشاؤها بشكل عملي وذلك بسبب كلفتها العالية والتي تعود إلى كثرة التوصيلات المطلوبة. ولكن يعد هذا النوع من أكثر الشبكات وثوقية ومواربة للأخطاء بسبب أن انهيار أي كبل سيتبعه وجود عدة طرق احتياطية بديلة.





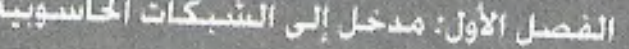
الشكل 1-10: طبولوجية كلية التشابك *(Full Mesh)*

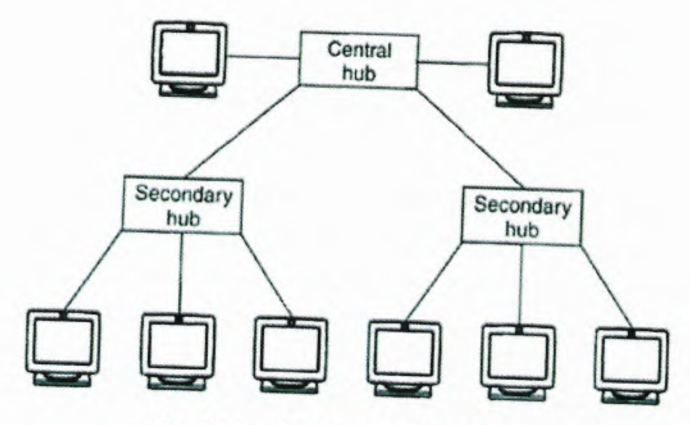
غالباً ما لا يستعمل هذا النوع من الشبكات في الشبكات المحلية بسبب كلفتها العالية، ولكن يستعمل عادة نسخة معدلة من هذا النوع تدعى بالشبكات التشابكية الهجينة *(Hybrid mesh)* تستخدم عند ربط بأنواع مختلفة من الشبكات المحلية في الشبكات الواسعة ومن ضمنها الانترنت.

في الشبكة التشابكية كاملة الوصل ذات عدد العقد n فإننا نحتاج إلى توصيلة.

5 - الشبكة الشجرية *(Tree)*

وتعتبر شكلاً آخراً من شبكة الناقل العمومية، وتسمى أيضاً بالشبكة الهرمية. توصل فيها عدة عقد بطريقة هرمية وعادة ما تكون العقدة الجذر هي مخدم عالي الأداء، أو حاسب مركزي أو جهاز مركزي مثل المبدلة ويسمى عادة بالرأس.



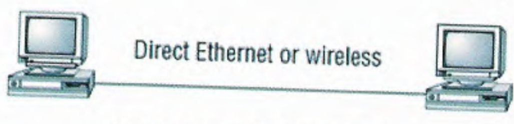


الشكل *1-11*: الطبولوجية الشجرية *(الهرمية)*

تعتبر الشبكات الشجرية مناسبة للمؤسسات متعددة المكاتب والتي تتصل جميعها بالمكتب المركزي. من أهم ميزات هذه الشبكات هي سهولة التوسع والإمكانية العالية لتحديد وعزل العقد المعطلة أو الفاشلة ما تسمح بقدرة عالية على استكشاف الأخطاء. ولكنها من ناحية أخرى تعاني من عقدة المركزية التي يسبب فشلها فشل الشبكة بالكامل.

6 - شبكات نقطة-لنقطة *(Point-to-Point)*

في هذا النوع من التوصيل يكون لدينا اتصال مباشر بين عقدتين على الشبكة *(مثل موجهين)* ما يحدد مسار اتصال وحيد بين عقدتين في الشبكة.



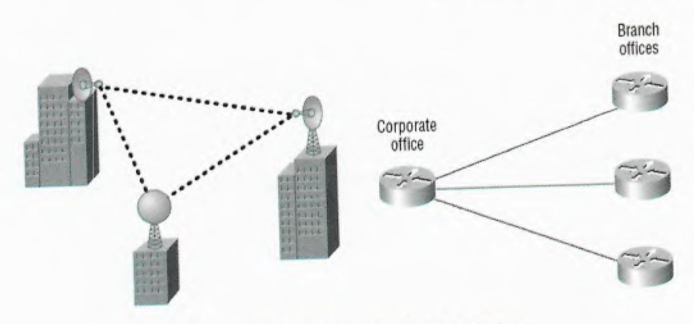
الشكل *1-12*: شبكات نقطة-لنقطة

يوجد الكثير من هذا النوع من التوصيل في الشبكات الواسعة. كذلك هناك الاستخدام الشائع لها بوصل خط اتصال لاسلكي مباشر بين جسرين لاسلكيين والمستعمل لوصل حواسيب في بناءين مختلفين.

في الشبكات المؤلفة من عدد من وصلات نقطة-لنقطة قد يتوجب على رسائل محددة السياق تسمى بالطرود *(Packets)* أن تمر عبر واحدة أو أكثر من العقد الوسطى. وبالتالي يمكن تواجد عدد من المسارات المختلفة بالطول والسعة؛ ولذلك فإن العثور على الطرق الفضلى هو مهم في شبكات نقطة-لنقطة. إن النقل نقطة-لنقطة الذي يتضمن مرسلاً وحداً ومستقبلاً وحيداً يسمى أحيانا ب *(Uncasting)*.

7 - شبكات نقطة-لمتعددة النقاط *(Point-to-Multipoint)* أو شبكات التعميم / البث *(Broadcasting Networks)*

وتتكون هذا الشبكات من عدة خطوط اتصال بين عقدة إلى عدة نقاط أو عقد في الشبكة.



الشكل *13-1*: شبكات نقطة-لمتعددة النقاط



وتكون في هذا النوع من الشبكات قناة الاتصال مشتركة بين كافة الأجهزة على الشبكة، وتستقبل الطرود المرسلة من أي جهاز من قبل كافة الأجهزة أخرى. عادة ما يتم تضمين حقل للعنوان في الطرود لتحديد المستقبل المحدد. وعند استقبال الطرد يقوم الجهاز بفحص حقل للعنوان فإذا كان هو الجهاز المستقبل المستهدف يقوم بمعالجة الطرود المستقبلة، أما إذا كانت بعنوان مخالف فيقوم بإهمالها فقط. الشبكات اللاسلكية هي مثال شائع عن هذه الأنظمة حيث يتم مشاركة الاتصالات على مساحة تغطية محددة من قبل الجهاز المرسل.

يمكن أن يتم تضمين عنوان خاص بهدف إيصال الرسالة إلى كافة المستخدمين ويسمى هذا النمط بالتعميم أو البث *(Broadcasting)*، أو إلى مجموعة جزئية محددة من المستخدمين والمعروفة بمتعددة الوجهات *(Multicasting)*.

1 - 4 - تمارين محلولة

1. ترمز *WAN* إلى:

a. *WAP Area Network*

b. <u>**Wide Area Network**</u>

c. *Wide Array Net*

d. *Wireless Area Network*

e. ولا إجابة من السابق.

2. عدد الأجهزة الأدنى لتشكيل شبكة حاسوبية هي:

a. 1

b. <u>2</u>

c. 3

d. لا يوجد

e. ولا إجابة من السابق.

3. إذا كانت كل الأجهزة تتصل إلى موزع مركزي فان الطبولوجيا تسمى:

a. *Bus*

b. *Ring*

c. <u>**Star**</u>

d. *Tree*

e. *Mesh*



4. تعتبر خدمة الرسائل الفورية *(instant messaging)* من النموذج:

a. مخدم-زبون

b. مخدم-مخدم

c. زبون-زبون

d. <u>ند-للند</u>

e. ولا إجابة من السابق.

5. ترمز *(PAN)* إلى:

a. *Personal Array Network*

b. *Protocol Area Network*

c. *Peer Area Network*

d. *Pan Area Network*

e. <u>Personal Area Network</u>

6. نمط الاتصال الذي يدعم معطيات في كلا الاتجاهين هو:

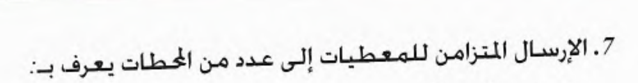
a. *Simplex*

b. <u>Duplex</u>

c. *Half Duplex*

d. *Multiplex*

e. ولا إجابة مما سبق.



7. الإرسال المتزامن للمعطيات إلى عدد من المحطات يعرف بـ:

a. **Broadcast**

b. *Aloha*

c. *Bandwidth*

d. *Analog transmission*

e. ولا إجابة مما سبق.

8. توصل الشبكات السلكية الشخصية *(PAN)* عن طريق:

a. *Blutooth*

b. *USB*

c. *Infrared*

d. *Firewire*

e. **كل من (b) و (d).**

9. تعتمد الشبكات المحلية *(LAN)* على -------- كأساس لنقل المعطيات:

a. *Multiplexing*

b. *Routing*

c. **Broadcasting**

d. *Flooding*

e. ولا إجابة من السابق.



***10*. ترمز *(ISP)* إلى:**

a. *Intra-domain Service Protocol*

b. *Internet Submission Protocol*

c. *Internet Service Provider*

d. *Inter-Section Protocol*

e. ولا إجابة من السابق.

***11*. يدعى الإجراء الذي يبادر إلى فتح الاتصال ______:**

a. المبادر

b. الزبون

c. الخدم

d. الند

e. ولا إجابة من السابق.

***12*. الشبكة التي لا يؤدي فشل أحد الكابلات فيها إلى فشل الشبكة بالكامل هي:**

a. *Ring*

b. *Bus*

c. *Star*

d. كل ما سبق

e. ولا إجابة مما سبق.



13. تستخدم تقنية *(Token Passing)* في الشبكات من النوع:

a. *Mesh*

b. ***<u>Ring</u>***

c. *Bus*

d. *Tree*

e. *Star*

14. أكثر أنواع الشبكات وثوقية هي:

a. *Bus*

b. *Star*

c. *Tree*

d. ***<u>Full Mesh</u>***

e. *Ring*

15. المعيار الأكثر شهرة للشبكات اللاسلكية هو:

a. *Ethernet*

b. *WiFi*

c. *IEEE 802.11*

d. كل من *(a)* و*(b)*

e. **<u>كل من *(b)* و*(c)*.</u>**



16. اختيار طبولوجية الشبكة يؤثر على:

a. نوع الأجهزة الحاسوبية المتصلة بالشبكة
b. نوع المعدات وإمكانياتها التي تحتاجها الشبكة.
c. مقدار نمو الشبكة مستقبلاً
d. كل ما سبق.
e. <u>كل من (b) و(c) فقط.</u>

17. تستخدم الشبكات الحلقية تقنية:

a. *Broadcasting*
b. <u>*Token Ring*</u>
c. *Multiplexing*
d. *Access point*
e. ولا إجابة مما سبق.

18. تعتبر شبكات نقل التلفزيون عبر الكبل من أكثر شبكات ----- شيوعاً:

a. *PAN*
b. *LAN*
c. <u>*MAN*</u>
d. *WAN*
e. ولا إجابة مما سبق.



19. تستخدم شبكات الناقل العمومي *(BUS)* لإنهاء ارتداد الإشارة:

a. *Switch*

b. *Hub*

c. ***Terminator***

d. *Free Token*

e. ولا إجابة مما سبق.



الفصل الثاني

البروتوكولات الشبكية

2 - 1 - المقدمة

ينتمي الكثير من العتاديات الصلبة إلى شركات مختلفة ذات مقاييس ومعايير متنوعة؛ مما يجعل عملية الاتصال والتواصل بين هذه التجهيزات المتنوعة معقدة جداً. لذلك كان على منتجي التجهيزات الحاسوبية والشبكية تبني واستخدام مجموعة من القواعد والاتفاقيات القياسية فيما بينهم. تسمى هذه المجموعة من القواعد والاتفاقيات التي تحكم تبادل البيانات بين طرفين على الشبكة بالبروتوكول. وهي كاللغة المشتركة المفهومة بين عدة أطراف يتكلمون بلغات مختلفة.

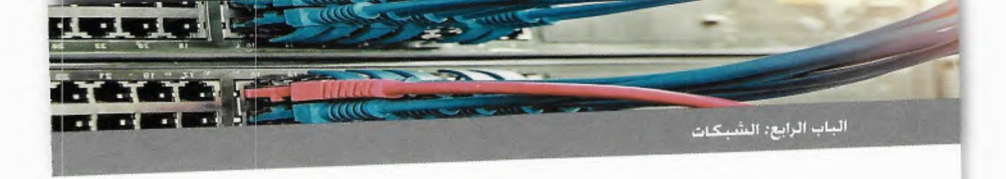
إن عملية الاتصال ضمن بيئة تعاونية بين مجموعة من التطبيقات وعلى أجهزة حاسوبية مختلفة هي معقدة جداً لتدار كوحدة واحدة. لذلك كان أيضاً من المهم وضع هيكلية وبنية لتعريف الاتصال مما يسمح بتطوير المعايير المناسبة.

2 - 2 - العناصر الأساسية بالبروتوكولات الشبكية هي:

1. **قواعد اللغة *(Syntax)*:** وتتضمن تنسيق البيانات *(Data format)*. الترميز ومستويات الإشارة.

2. **الدلالات *(Semantics)*:** وتتضمن أشياء مثل معلومات التحكم الخاصة بالتعاون والتعامل مع الأخطاء.

3. **التوقيت *(Timing)*:** وتتضمن تطابق سرعات النقل وتسلسل البيانات *(Sequencing)*.



في الشبكات الحاسوبية تحتاج الطرفيات الموجودة على مضيفين *(Hosts)* مختلفين للتواصل فيما بينهم. الطرفية *(Entity)* يمكن أن تكون تطبيقات لمستخدمين، أو حزم نقل ملفات، أو أنظمة إدارة قواعد بيانات. بالتعريف عموماً يمكن القول أن الطرفية هي أي شيء قادر على إرسال أو استقبال البيانات في الشبكة.

2 - 3 - وظائف البروتوكولات الشبكية

يمكن تصنيف عمل البروتوكولات الشبكية بالأقسام التالية:

- التقسيم *(Segmentation)* وإعادة التشكيل *(Reassembly)*.
- التغليف *(Encapsulation)*.
- التحكم بالاتصال *(Connection Control)*.
- التسليم المرتب *(Ordered Delivery)*.
- التحكم بالدفق *(Flow control)*.
- التزامن *(Synchronization)*.
- العنونة *(Addressing)*.
- التنخيب *(Multiplexing)*.
- خدمات الإرسال.

التقسيم وإعادة التشكيل: يتم عادة تقسيم الرسائل إلى تسلسل من كتل محدودة الحجم من البيانات. على البروتوكولات في الطبقات الدنيا تقسيم البيانات إلى كتل ذات أحجام محدودة أصغر. وتسمى هذه

العملية بالتقسيم *(Segmentation)*، أو التجزئة *(Fragmentation)*. تسمى الكتل الناجّة عن التقسيم بوحدة معطيات البروتوكول *(Protocol Data Unit - PDU)*. العملية المعاكسة للتقسيم هي إعادة التشكيل في الطرف الآخر *(Reassembling)*.

التغليف: تحتوي كل وحدة معطيات بروتوكول *(PDU)* إضافة إلى بيانات المرسل معلومات تحكم أيضاً. حتى أن بعض الوحدات تحوي فقط معلومات تحكم دون بيانات. عملية التغليف هي إضافة هذه المعلومات إلى البيانات المرسلة إما عبر ترويسة *(Header)* و/أو عبر لاحقة *(Trailer or Footer)*.

ويمكن تصنيف معلومات التحكم بثلاثة أقسام:

- العنوان: ويتضمن عنوان المرسل و/أو عنوان المستقبل.
- ترميزات تحكمية خاصة بكشف الأخطاء.
- التحكم بالبروتوكول.

التحكم بالاتصال: يمكن أن تبدأ طرفية بالاتصال دون التنسيق أو التعاون مع الطرفية المقابلة وهو ما يسمى بالنقل عديم التوجيه *(Connectionless)*. أما في حالة التعاون المسبق للإرسال بين المرسل والمستقبل فيتم تأسيس اتصال منطقي بين الطرفيتين؛ يسمى هذه الاتصال بالاتصال الموجه *(Connections-Oriented)*. وهو يتضمن ثلاثة مراحل: إنشاء الاتصال، ومن ثم نقل البيانات، وأخيراً إنهاء الاتصال.

التسليم المرتب: بعد تقسيم الرسائل إلى وحدات أصغر يمكن لهذه الوحدات *(PDUs)* وخلال الانتقال في الشبكة أن لا تصل بذات الترتيب الذي أرسلت فيه؛ ولذلك بالأخص في بروتوكولات ذات الاتصال الموجه يكون من المطلوب أن تصل بذات ترتيب الإرسال.

التحكم بالتدفق: وهو عملية تنفذ من قبل طرفية الاستقبال لتحديد كمية أو معدل إرسال البيانات من قبل الطرفية المرسلة. عادة ما يتم تحقيق هذه الوظيفة في عدة بروتوكولات.

التحكم بالخطأ: وهي التقنيات المستخدمة لضمان عدم حدوث أخطاء على البيانات أو معلومات التحكم أو عدم ضياعها. تتضمن أغلب التقنيات مهمة الكشف عن الأخطاء وإعادة إرسال الطرود. أيضاً التحكم بالخطأ هو عملية عليها أن تنفذ في مستويات مختلفة من البرتوكولات.

التزامن: يكون من المهم في بعض الأحيان أن تكون طرفيتي اتصال في حالة معرفة تماماً مثلاً من أجل نقاط الفحص *(Check Points)* أو الإنهاء *(Termination)*. وهو ما يسمى بالتزامن.

العنونة: حتى تستطيع طرفيتين الاتصال باستخدام وصلات غير وصلات نقطة-لنقطة، عليها أن تعرّف الوجه المرسلة إليها ومن هو المرسل.

التنخيب *(Multiplexing)*: يسمح التنخيب بإجراء عدة اتصالات في نفس الوقت.

خدمات الإرسال: قد يقدم البروتوكول مجموعة من الخدمات إلى الطرفيات التي تستخدمه. الأمثلة الأكثر شيوعاً من الخدمات هي:

- تحديد الأولوية *(Priority)*.
- مستوى الخدمة *(Level of Service)*: قد تتطلب بعض أنواع الاتصالات حدوداً قصوى معينة من التأخير، أو معدل نقل بيانات أصغري على البروتوكول تأمينها لها.
- السرية.



2 - 4 - النموذج المرجعي الطبقي (*OSI*)

2 - 4 - 1 - تعريف النموذج المرجعي

عرفت منظمة المعايير العالمية (*International Standards Organization - ISO*)، وهي منظمة دولية متخصصة بوضع المعايير التي تحظى بالقبول في شتى أنحاء العالم، نموذجاً لترابط الأنظمة المفتوحة (*Open Systems Interconnection - OSI*) عام *1970*. يقصد بالنظام المفتوح مجموعة البروتوكولات التي تسمح بتحقيق الاتصال بين أي نظامين مختلفين بغض النظر عن بنيتيهما الداخلية. ضمن هذا السياق، يهدف نموذج *OSI* إلى تسهيل عملية الاتصال بين الأنظمة المختلفة دون الحاجة إلى إجراء تعديلات على مكوناتها العتادية أو البرمجية.

إن النموذج *OSI* ليس بروتوكولاً بحد ذاته وإنما نموذجاً لفهم وتصميم الشبكات تصميماً مرناً ومتيناً وقابلاً للتعامل مع شبكات أخرى. يتألف النموذج (*OSI*) من سبع طبقات منفصلة ومتكاملة مع بعضها البعض، تسمح كل طبقة بتحقيق جزء من عملية نقل المعلومات عبر الشبكة. كان من المتوقع أن يصبح هذا النموذج هو النموذج المعتمد علمياً وتجارياً لكن لم يتم تحقيقه كمنتج تجاري وإنما يستخدم كأداة ومرجع علمي.

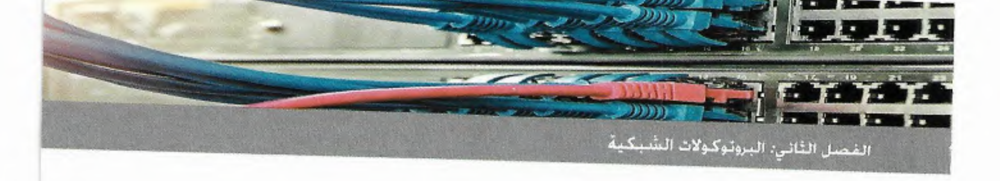
تمر العملية الكاملة لنقل البيانات على الشبكة بمجموعة من الخطوات، وفي كل خطوة تنفذ مهام محددة لا يمكن تنفيذها في خطوة أخرى، ولكل خطوة بروتوكول محدد، أو مجموعة من البروتوكولات في طبقة محددة، يحدد كيفية تنفيذ المهام المتعلقة بهذه الخطوة، و تكون هذه الخطوات متشابهة في كل جهاز على الشبكة، ويجب ملاحظة أن الجهاز المرسل يقوم بإتباع هذه الخطوات من الأعلى إلى الأسفل بينما يقوم الجهاز المستقبل بإتباع هذه الخطوات بشكل معكوس من الأسفل إلى الأعلى.



يقسم هذا النموذج وظائف الشبكات الحاسوبية إلى طبقات سبع هي على الترتيب ومن الأسفل:

1. الطبقة الفيزيائية *(The Physical Layer)*.
2. طبقة ربط البيانات *(The Data Link layer)*.
3. طبقة الشبكة *(The Network layer)*.
4. طبقة النقل *(The Transport layer)*.
5. طبقة الجلسة *(The Session layer)*.
6. طبقة التقديم *(The Presentation layer)*.
7. طبقة التطبيق *(The Application layer)*.

تقدم كل طبقة خدمات للطبقات الأعلى منها بينما تستفيد من خدمات الطبقات الأسفل. لذلك فإن وحدة معطيات بروتوكول *(PDU)* ناجمة عن طبقة ما *N*، عليها أن تخدم عندما تنتقل إلى طبقة أدنى *N-1* وتسمى وحدة معطيات خدمية لتلك الطبقة *(Service Data Unit – SDU)*. الطبقات الثلاث السفلى مخصصة لنقل البتات من البيانات وتبادلها بين الشبكات المختلفة. أما الطبقات الثلاث العليا فهي مخصصة لتطبيقات وبرامج المستخدم. تعمل الطبقة الوسطى كواجهة بين الطبقات السفلى والعليا. يفصل بين كل طبقة وأخرى واجهة تخاطبية *(Interface)* تعمل على تمرير البيانات والمعلومات بين الطبقات. وتكون هذه الواجهة معرفة بعناية وبشكل مستقل عن التحقيق وذلك بغية تمكين تغيير تحقيق الطبقة بدون تغيير الطبقات المجاورة.



2 - 4 - 2 - طبقات النموذج *OSI*

1 - الطبقة الفيزيائية *(Physical Layer)*

تهتم الطبقة الفيزيائية بنقل البتات الخام على وسيط نقل. وتتضمن واجهة التخاطب الفيزيائية بين الأجهزة والقواعد التي على أساسها تمرر البتات من جهاز إلى آخر. يجري تصميم هذه الطبقة على أساس أنه عندما يرسل المصدر خانة الواحد المنطقي فإن الوجهة سيستقبل الواحد المنطقي وليس الصفر. لذلك فهي تُعنى بالمواصفات الكهربائية والإلكترونية والميكانيكية لأوساط الاتصال.

		Layer
UPPER LAYERS	7	**Application Layer** ✓ Message format, Human-Machine Interfaces
	6	**Presentation Layer** ✓ Coding into 1s and 0s; encryption, compression
	5	**Session Layer** ✓ Authentication, permissions, session restoration
TRANSPORT SERVICE	4	**Transport Layer** ✓ End-to-end error control
	3	**Network Layer** ✓ Network addressing; routing or switching
	2	**Data Link Layer** ✓ Error detection, flow control on physical link
	1	**Physical Layer** ✓ Bit stream: physical medium, method of representing bits

الشكل *2-1*: طبقات النموذج المرجعي *(OSI)* وأهم الخدمات في كل طبقة

وتكون المهمات الرئيسية في هذه الطبقة هي:

- تعريف مواصفات العتاد الصلب: وتضم تفاصيل الكابلات وعملها، الوصلات، مرسلات ومستقبلات الإشارة الراديوية اللاسلكية، بطاقات واجهة التخاطب الشبكية *(Network Interface Card –NIC)*.

- الترميز وإرسال الإشارة: هذه الطبقة مسؤولة عن عمليات الترميز والإشارة التي تحول المعطيات من بتات إلى إشارات كهربائية أو ضوئية حسب نوع وسيط النقل يمكن لها أن تنتقل على الشبكة.
- إرسال المعطيات واستقبالها: هذه الطبقة هي من تقوم عملياً بإرسال واستقبال الإشارات وتحديد نوع الإرسال على الخط (مزدوج *(Duplex)* أو بسيط *(Simplex)*).
- طريقة تأسيس الاتصال البدائي وطريقة قطعه والمدة الزمنية التي يستغرقها إرسال البت.
- تحديد الطبولوجية الفيزيائية للشبكة.
- إعداد الخط: هل الوصلة ستكون نقطة لنقطة أو متعددة النقاط.

لا تضيف الطبقة الفيزيائية أي ترويسة للأطر القادمة من طبقة وصلة المعطيات.

2 - طبقة وصلة المعطيات *Data Link Layer*

بينما توفر الطبقة الفيزيائية خدمات نقل البتات الخام فقط، تحاول طبقة وصل المعطيات جعل الوصلة الفيزيائية موثوقة وتوفر الوسائل التي من خلالها يمكن تفعيل، والحفاظ على، وتعطيل الخط. تهدف هذه الطبقة إلى حجب والتحكم بجميع مشاكل وأخطاء الطبقة الفيزيائية وتزويد طبقة الشبكة بخدمة نقل خالية من الأخطاء. **لذلك تعمل على تأمين خدمة موثوقة نقطة- لنقطة عن طريق تحقيق ما يلي:**

- **التأطير *(Framing)*:** عن طريق تحويل البتات المستقبلة من الطبقة الفيزيائية إلى وحدات معطيات قابلة للإدارة تدعى بالأطر *(Frames)*.

- **العنونة الفيزيائية:** عندما تريد محطة ما إرسال إطار إلى وجهة تنتمي إلى نفس الشبكة فإن طبقة وصلة المعطيات تضيف ترويسة للتعريف عن المصدر والوجهة. أما إذا كانت الوجهة تنتمي إلى شبكة أخرى فيجري وضع العنوان الفيزيائي للموجّه الافتراضي.
- **التحكم بالتدفق *(Flow Control)*:** إذا كانت سرعة استهلاك المعطيات لدى الوجهة أقل من سرعة إنتاج المعطيات لدى المصدر، فإن طبقة وصلة المعطيات تفرض استخدام آلية تحكم بالتدفق لتجنب إغراق الوجهة.
- **التحكم بالأخطاء *(Error Control)*:** تضيف طبقة وصلة المعطيات نوعاً من الوثوقية عن طريق إضافة آلية لاكتشاف ولإعادة إرسال الأطر المشوهة أو الضائعة. كما يمكنها أيضاً التعرف على الأطر المكررة.
- **التحكم بالنفاذ *(Access Control)*:** عندما يتم ربط جهازين أو أكثر إلى وسيط نقل مشترك فإنه يتوجب على طبقة وصلة المعطيات اختيار أي جهاز سيرسل في كل لحظة.

تقسم هذه الطبقة عادة إلى قسمين:

- طبقة دنيا هي التحكم بالوصلة المنطقية *(Logical Link Control – **LLC**)*: وتعرّف العمليات المطلوبة لتأسيس والتحكم بالوصلات المنطقية بين التجهيزات المحلية على الشبكة. معظم تقنيات الشبكات المحلية تستخدم بروتوكول *IEEE 802.2 LLC*.
- طبقة عليا هي التحكم بالنفاذ إلى الوسط *(Media Access Control-**MAC**)*: وتتضمن عمليات التحكم بالنفاذ إلى الوسط وتجنب التصادمات. مثلاً تستخدم شبكة الإيثرنت طريقة *(CSMA / CD)* للتحكم بالنفاذ إلى الوسط. بينما تستخدم شبكة *(Token Ring)* تمرير العلامة *(Token Passing)* للتحكم بالنفاذ.



تجدر الإشارة هنا إلى أن طبقة وصلة المعطيات تضيف ترويسة ولاحقة للطرود القادمة من طبقة الشبكة.

3 - طبقة الشبكة *(Network Layer)*

تهتم هذه الطبقة بطريقة توجيه الطرود من المصدر إلى الوجهة (مضيف -إلى-مضيف) بعد اجتياز مجموعة من الشبكات الوسيطة (أي اختيار أفضل طريق بين نقطتين). **لذلك تقوم بتحقيق ما يلي:**

- **العنونة المنطقية** *(Logical Addressing)*: يملك كل جهاز موصول إلى شبكة ما عنوانين: الأول فيزيائي محلي والثاني منطقي شامل على جميع الشبكات.
- **التوجيه** *(Routing)*: عندما يجري ربط أكثر من شبكة معاً لتشكيل ترابط شبكات فإنه يتوجب على أجهزة الربط (الموجهات أو المبدلات) توجيه الطرود عبر الشبكات حتى تصل إلى الوجهة النهائية.
- تحديد كون المسار هو معرف مسبقاً أو يحسب في حينه حسب ظروف الشبكة *(Static / Dynamic)*.
- **التجزئة وإعادة التشكيل** *(Segmentation and Reassembly)*: تجزئة الرسائل الطويلة إلى طرود قصيرة وإعادة تجميعها لدى الوجهة.
- **معالجة الاختناقات** *(Congestion Control)*: والتي يمكن أن تحدث في أوقات الذروة.

يكون الجهاز المرسل في هذه الطبقة منشغلاً بحوار مع الشبكة لتحديد عنوان الوجهة، ولطلب بعض التسهيلات الشبكية مثل الأولوية *(Priority)*.



4 - طبقة النقل *(Transport Layer)*

تضمن هذه الطبقة أن يتم تسليم وحدات المعطيات *(segment)* إلى وجهتها دون أخطاء، وبالترتيب الأصلي لها، ودون أي ضياعات أو تكرار. وهي تعمل كوسيط بين المستخدم *(الطبقات الثلاث الأولى)* وبين شبكة الاتصال. تؤمن هذه الطبقة نقل الرسالة بكاملها نقلاً موثوقاً وهي من نوع نهاية - لنهاية *(End-to-End)*. لاحظ أن الفرق بين طبقتي الشبكة والنقل يكمن في كون الأولى تعالج كل طرد على حدا، وتنقله بشكل مستقل عن بقية الطرود دون معرفة العلاقة بين الطرود؛ بينما تضمن طبقة النقل وصول كامل الرسالة وصولاً سليماً خالياً من الأخطاء ومرتباً بعد الإشراف على عمليات التحكم بالأخطاء والتدفق بين المصدر والوجهة.

المهام الأساسية لطبقة النقل

- **عنونة نقاط الخدمة** *(Service Point Addressing)*: لا تقتصر عملية توصيل المعطيات على الوصول إلى عنوان الوجهة فقط؛ وإنما تعني أيضاً الوصول إلى الإجراء *(أي البرنامج التطبيقي)* المطلوب. يجري تحديد عنوان الإجراء المصدر أو الوجهة عن طريق معرف نقطة نفاذ الخدمة *(Service Access Point - SAP)* أو رقم البوابة *(Port Number)*.

- **التجزئة وإعادة التشكيل** *(Segmentation and Reassembly)*: نقصد بذلك إمكانية أن تقوم طبقة النقل بتقسيم الرسالة المطلوب إرسالها إلى أجزاء قابلة للنقل وإضافة أرقام تسلسلية إلى هذه الأجزاء لتستطيع المحطة الوجهة إعادة تجميع الأجزاء واستخلاص الرسالة الأصلية.

- **التحكم بالوصلة *(Connection Control)*:** يمكن أن تكون طبقة النقل ذات اتصال موجّه *(Connection-oriented)* أو عديم التوجيه *(Connectionless)*. عندما تكون طبقة النقل عديمة التوجيه فإنها تعالج كل مقطع *(Segment)* (والمقطع هنا هو وحدة المعطيات على مستوى طبقة النقل) من رسالة على أنه مقطع مستقل، وتوصله إلى طبقة النقل لدى الوجهة. بينما تؤسس طبقة النقل ذات الاتصال الموجه اتصال مع طبقة النقل لدى الوجهة قبل البدء بتراسل الطرود ويجري قطع الارتباط عند الانتهاء.

- **التحكم بالتدفق:** كما هو عليه الحال ضمن طبقة وصلة المعطيات فإن طبقة النقل تقوم بالتحكم بالتدفق لكن من نهاية لنهاية وليس من نقطة لنقطة.

- **التحكم بالأخطاء:** كذلك الأمر بالنسبة للتحكم بالأخطاء الذي يجري أيضاً من نهاية لنهاية.

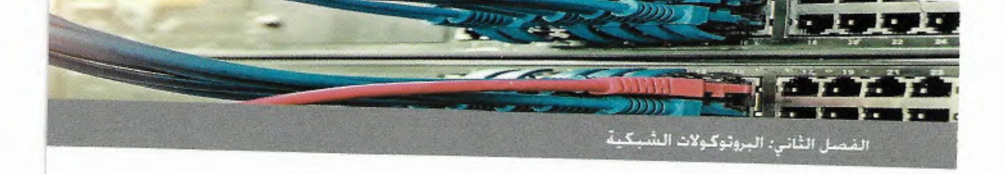
5 - طبقة الجلسة *(Session Layer)*

تسمح طبقة الجلسات لمستثمرين موجودين على محطات مختلفة بتأسيس جلسات فيما بينهم. **تؤمن الجلسات مجموعة من الخدمات مثل:**

- **إدارة الحوار *(Dialog Control)*:** تسمح طبقة الجلسات لنظامين بالدخول ضمن حوار، فهي تسمح بتحقيق التواصل بين إجراءين بشكل نصف مزدوج *(Half Duplex)* أو مزدوج.

- **تشكيل المجموعات *(Grouping)*:** يمكن تعليم تدفق المعطيات لتحديد كمجموعات من المعطيات.

- **التزامن وتحديد نقاط الفحص *(Check Pointing)*:** تسمح طبقة الجلسة لإجراء ما بإضافة نقاط تزامن إلى سلسلة معطيات؛ فإذا



وقع عطل ما أثناء الإرسال يمكن العودة إلى آخر نقطة تزامن *(Checkpoint)* بدلاً من إعادة الإرسال من جديد.

6 - طبقة التقديم *(Presentation Layer)*

تهتم هذه الطبقة بقواعد التخاطب *(Syntax)* ودلالات *(Semantic)* المعلومات المتراسلة بين نظامين. المهمة الرئيسية لهذه الطبقة هي حل الاختلافات في شكل المعطيات. نذكر من المهام الأساسية لطبقة التقديم:

- الترجمة *(Translation)*: تسمح طبقة التقديم بتحقيق الترجمة بين ترميزات مختلفة للمعلومات مثل *(ASCII)* و *(EBCDIC)*.
- التعمية/التشفير *(Encryption)*.
- ضغط البيانات *(Compression)*.

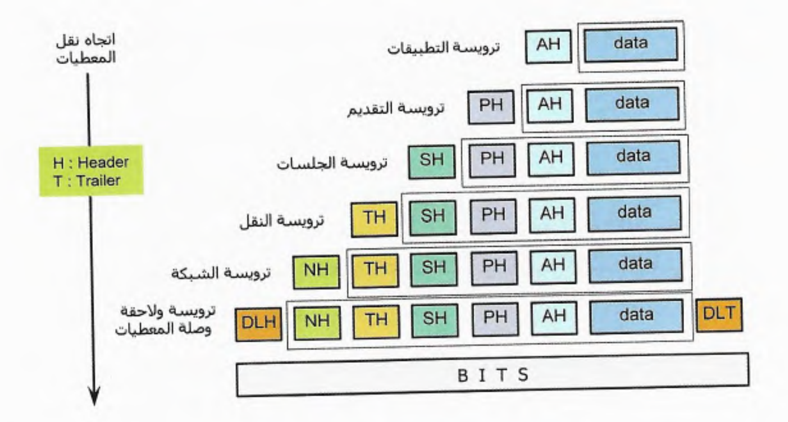
7 - طبقة التطبيقات *(Application Layer)*

تحوي طبقة التطبيقات مجموعة متنوعة من البروتوكولات التي يحتاجها المستثمرون بكثرة. من التطبيقات واسعة الانتشار نذكر منها *(Hyper Text Transfer Protocol - HTTP)* الذي يشكل أساس الشبكة العنكبوتية العالمية *(WWW)*. عندما يريد متصفح صفحة ويب فإنه يرسل اسم الصفحة إلى مخدم الويب باستخدام *(HTTP)* الذي يرسل الصفحة للمستثمر.

نذكر من التطبيقات الشائعة المستخدمة في هذه الطبقة:

- البريد الإلكتروني *(e-mails)*.
- نقل الملفات والنفاذ إليها وإدارتها.
- برمجيات البحث واستخلاص المعلومات.





الشكل *2-2*: تغليف الرسالة ضمن الطبقات

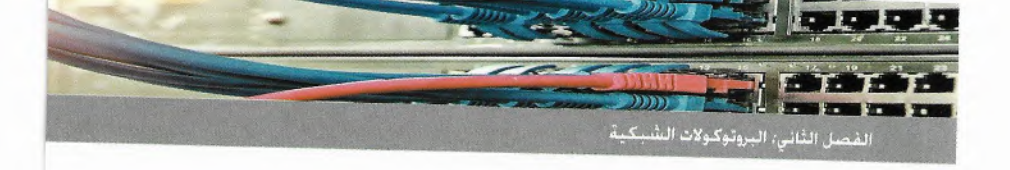
2 - 5 - النموذج الشبكي *(TCP /IP)*

2 - 5 - 1 - المقدمة

من أكثر البروتوكولات رواجاً هو مجموعة بروتوكولات *(Transmission Control Protocol / Internet Protocol - TCP / IP)*.
وهو يتضمن مجموعة من البروتوكولات الجزئية التي تسمح للشبكات بالاتصال مع الانترنت، أو الاتصال مع بعضها البعض لتشكل شبكات انترانت خاصة. أنشأ من قبل وكالة مشاريع الأبحاث الدفاعية المتقدمة *(DARPA)*. ومعظم أنظمة التشغيل الحالية تستخدم حزمة بروتوكولات *(TCP/IP)* كالبروتوكول الافتراضي للاتصال مع الشبكة.

2 - 5 - 2 - طبقات حزمة البروتوكول *(TCP/IP)*

تقسم حزمة بروتوكولات *(TCP/IP)* إلى أربع طبقات يمكن أن تتقاطع مع النموذج المعياري *(OSI)*، **وهذه الطبقات الأربع هي:**



1 - **طبقة التطبيق *(Application Layer)***

وهي تقابل تقريبا طبقات التطبيق والجلسة والتقديم من النموذج المرجعي *(OSI)*. يمكن للتطبيقات في هذه الطبقة أن تنفذ إلى الشبكة باستخدام بروتوكولات مثل *(HTTP)* وبروتوكول نقل الملفات *(File Transfer Protocol – **FTP**)* أو بروتوكول الزمن *(Network Time Protocol – **NTP**)* أو بروتوكول تهيئة المضيف ديناميكياً *(Dynamic Host Configuration Protocol – **DHCP**)*. والكثير غيرها.

2 - **طبقة النقل *(Transport Layer)***

وهي تقابل تقريباً طبقة النقل في النموذج المرجعي *(OSI)*. تتضمن هذه الطبقة بروتوكول التحكم بالإرسال *(Transmission Control Protocol – **TCP**)* وبروتوكول وحدة معطيات المستخدم *(User Datagram protocol – **UDP**)* والتي تقدم التحكم بالتدفق، فحص الأخطاء، وترتيب الطرود.

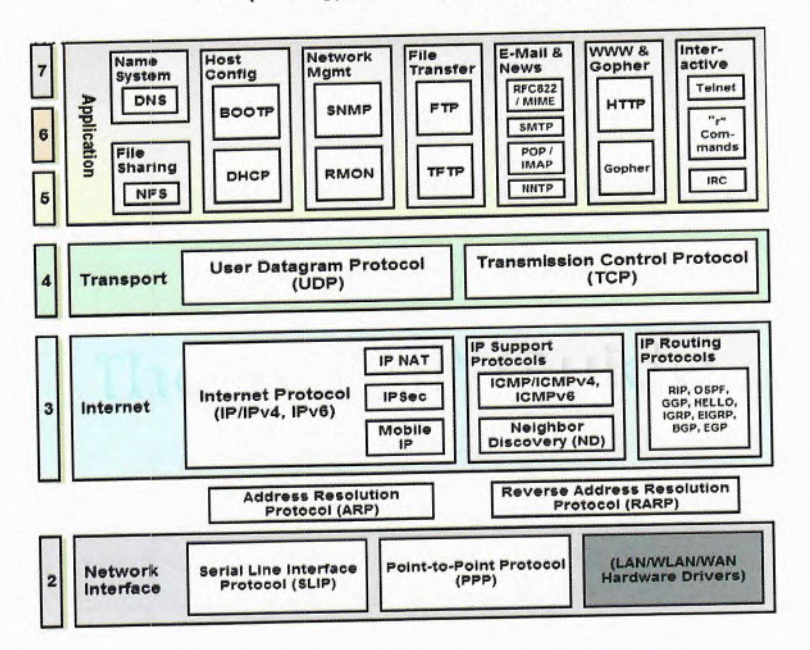
3 - **طبقة الانترنت *(Internet Layer)***

وهي تقابل طبقة الشبكة في النموذج المرجعي *(OSI)*. تضم هذه الطبقة بروتوكول الانترنت *(Internet Protocol – **IP**)*. وكذلك بروتوكول رسائل التحكم بالإنترنت *(Internet Control Message Protocol – **ICMP**)*، الذي هو بروتوكول داعم يستخدم *(IP)* لإيصال معلومات التحكم المتعلقة بحصول خطأ أثناء نقل طرود *(IP)* وهو يحتوي على رسائل من أشهرها التي تأتي مع الأداة *(Ping)*. وبروتوكول حل العنوان *(Address Resolution Protocol - **ARP**)* الذي يترجم عنوان انترنت *(IP)* إلى عنوان شبكة فيزيائي مثل عناوين الإيثرنت، وبروتوكول حل العنوان العكسي *(Reserve Address Resolution Protocol – **RARP**)* الذي يؤدي المهمة المعاكسة أي يحول عنوان شبكة فيزيائي إلى عنوان انترنت *(IP)*.



4 - طبقة واجهة الشبكة *(Network Interface Layer)* أو طبقة الوصلة *(Link Layer)*

وهي تقابل تقريباً الطبقتين الفيزيائية وطبقة وصلة المعطيات في النموذج المرجعي للشبكات *(OSI)*. تعالج هذه الطبقة مسائل تنسيق المعطيات وإرسالها إلى الواجهة التخاطبية للشبكة.



الشكل 2-3: حزمة بروتوكولات *(TCP/IP)* وعلاقتها بالنموذج المرجعي *(OSI)*

يمثل البروتوكولان *(UDP / TCP)* طبقة النقل في النموذج *(TCP / IP)*. البروتوكول *(UDP)* هو بروتوكول بسيط يؤمن وظائف نقل غير موثوقة الأمر الذي سبب اعتماد معظم التطبيقات على خدمات *(TCP)* التي تؤمن الموثوقية من نهاية لنهاية.



يُسَلِّم بروتوكول الانترنت *(IP Protocol)* طرود المعطيات من المضيف المصدر إلى المضيف الوجهة، حيث تعرف عملية التسليم هذه ببروتوكول مضيف إلى مضيف *(Host-to-Host Protocol)*. من الممكن أن ينفذ المضيف المستقبل للمعطيات عدة عمليات مختلفة ومتزامنة.

تُعَرِّف بروتوكولات طبقة النقل في النموذج *(TCP / IP)* مجموعة من الوصلات المفاهيمية للإجراءات المستقلة *(process)*، تدعى ببروتوكول المنافذ أو البوابات *(Ports)*. المنفذ هو نقطة وجهة، وعادة ما يكون عبارة عن صوان *(Buffer)*، لتخزين المعطيات، من أجل استخدامها من خلال إجراء ما. تُمْنَح الوصلة ما بين الإجراءات والمنافذ المتوافقة معها من قبل نظام التشغيل.

بروتوكول الانترنت *(IP)* هو بروتوكول مضيف إلى مضيف، يمكن أن يسلم الطرود من جهاز فيزيائي إلى آخر. بينما بروتوكولات طبقة النقل، فهي بروتوكولات إجراء إلى إجراء أو منفذ إلى منفذ، والتي تعمل فوق بروتوكولات *IP* من أجل توصيل الطرود إلى خدمات بروتوكول الانترنت *IP* عند بداية النقل، ومن خدمات بروتوكول الانترنت إلى المنفذ الوجهة عند نهاية النقل. وهذا ما يبينه الشكل *(4-2)*.

telnet (client)
(51001)
TCP or UDP
IP
Data Link
physical

telnet (server)
(23)
TCP or UDP
IP
Data Link
physical

الشكل *4-2*: عناوين المنفذ

عناوين المنفذ

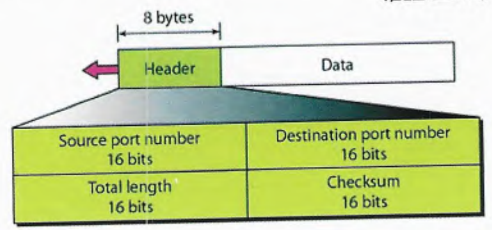
يُعَرَّف كل منفذ من خلال عنوان، وهو عبارة عن رقم صحيح موجب موجود ضمن ترويسة طرد طبقة النقل. تستخدم طرود معطيات بروتوكول الانترنت *(IPv4)* في النسخة الرابعة منها عنوان انترنت مؤلف من *32* خانة لكل مضيف. يستخدم الطرد على مستوى طبقة النقل عنوان للمنفذ مؤلف من *16* خانة وهي كافية للسماح بدعم حتى *65536* منفذ.

2 - 6 - بروتوكول وحدة معطيات المستخدم *(UDP)*

وهو بروتوكول لنقل غير الموثوق غير موجّه. لا يضيف هذا البروتوكول أي خدمات إضافية على بروتوكول الانترنت *(IP)* باستثناء توفير اتصال من نوع إجراء لإجراء بدلاً من مضيف لمضيف. يقوم باختبار أخطاء محدود (باستخدام اختبار المجموع *(Checksum)*. لا يضمن تسلسل الطرود. ولا يحدد الطرود التالفة عندما يستلم تقريراً بوجود خطأ.

رسالة معطيات المستخدم *(User datagram)*

يدعى الطرد الذي يولده بروتوكول *(UDP)* برسالة المعطيات *(User Datagram)*. تحتوي رسالة المعطيات على ترويسة ذات حجم ثابت (*8* بايت) بالإضافة إلى المعطيات. يبين الشكل *(2-5)* بنية رسالة معطيات المستخدم.



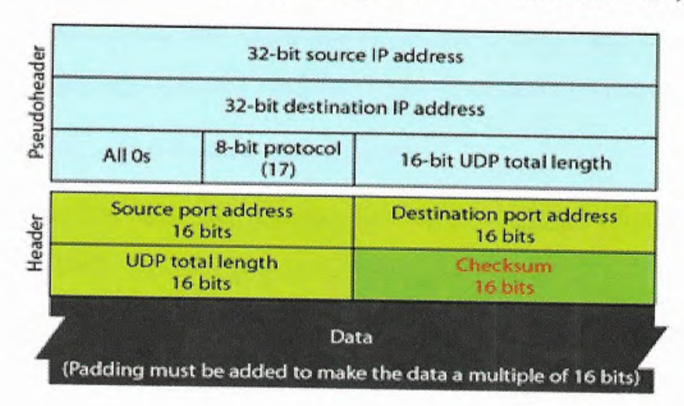
الشكل *2-5*: بنية رسالة معطيات المستخدمة في بروتوكول *(UDP)*



نوضح فيما يلي حقول ترويسة رسالة المعطيات:

- **رقم البوابة المصدر** ***(Source Port Number)***: يستخدم هذا الرقم من خلال الإجراء المنَفذ على المضيف المنشئ للرسالة، وهو بطول *16* بت.
- **رقم البوابة الوجهة** ***(Destination port number)***: يستخدم هذا الرقم من خلال الإجراء المنفَذ على المضيف المستقبل للرسالة وهو بطول *16* بت أيضاً.
- **الطول الكلي** ***(Total length)***: حقل مؤلف من *16* بت يُحَدِد الطول الكلي لرسالة المعطيات *(*الترويسة مع البيانات*)*.
- **اختبار المجموع** ***(Checksum)***: قيمة تحقق من عدم حدوث أخطاء في رسالة المعطيات.

يسبق الترويسة شبه ترويسة تضم عنوان المرسل والمستقبل إضافة إلى حقول أخرى كما هو مبين في الشكل *(6-2)*.



الشكل *6-2*: شبه الترويسة في *UDP*

من الممكن أن تصل رسالة بشكل آمن وسليم حتى ولو كان مجموع الاختبار لا يحتوي على شبه الترويسة. لكن إذا تشوهت ترويسة *IP* فمن الممكن أن تصل الرسالة إلى الوجهة الخطأ. أضيف الحقل *(Protocol)* للتأكد من أنَّ الطرد يخص بروتوكول *(UDP)*. قيمة هذا الحقل من أجل بروتوكول *(UDP)* هي *17*. إذا تغيرت هذه القيمة خلال عملية النقل فإنَّ اختبار المجموع عند المستقبل سيكتشف هذا التغيير وسيقوم بروتوكول *(UDP)* بإسقاط الطرود، ولن يُسَلِّمُ إلى البروتوكول الخطأ.

استخدامات بروتوكول *(UDP)*

يعتبر بروتوكول *UDP* مناسباً من أجل بعض التطبيقات مثل:

- الإجرائيات التي تتطلب بساطة في طلب ورد الاتصال، مع تحكم محدود بالأخطاء.
- العمليات التي تمتلك تقنيات التحكم بالأخطاء والتدفق داخلياً، مثل بروتوكول نقل الملفات العادي *(Trivial File Transfer protocol – **TFTP**)*.
- النقل متعدد الوجهات *(Multicasting)*.
- عمليات الإدارة، مثل بروتوكول إدارة الشبكة البسيط *(Simple Network Management Protocol - **SNMP**)*.
- بعض بروتوكولات تحديث التوجيه، مثل بروتوكول معلومات التوجيه *(Routing Information Protocol – **RIP**)*.



2 - 7 - برتوكول التحكم بالإرسال *(TCP)*

يعمل على طبقة النقل ويقدم خدمات توصيل معطيات موثوقة. وهو بروتوكول من نوع الاتصال الموجه. **ويقوم بالعمليات التالية:**

- يقوم بتقسيم الرسائل الطويلة إلى طرود أصغر تدعى في هذا البروتوكول بالمقطع *(Segment)*. يتضمن كل مقطع رقماً تسلسلياً *(SN)* من أجل إعادة الترتيب بعد الاستلام، ورقم تعريف *(ID)*، وحقلاً يحدد حجم نافذة الاستقبال.
- يؤسس الاتصال بين عقدتين على الشبكة قبل أن يبدأ البروتوكول بإرسال البيانات.
- يضمن توصيل معطيات موثوق عبر استخدام تقانات التسلسل *(Sequencing)* وفحص الأخطاء *(Checksums)* وتقنية إعادة إرسال الأطر التالفة.
- يقدم التحكم بالتدفق لضمان عدم إغراق عقدة بالمعطيات في حال أن المرسل أسرع من قدرة المستقبل على معالجة البيانات الواردة.
- تغلف المقاطع في رسالة معطيات طبقة *IP* وترسل. وعند نهاية الاستقبال، يجمع بروتوكول *(TCP)* جميع طرود المعطيات كما استقبلت ويعيد ترتيبها بالاعتماد على الأرقام التسلسلية.

اتصال بروتوكول التحكم بالنقل

عندما يريد إجراء عند الموقع *A* أن يرسل ويستقبل معطيات من إجراء أخرى عند الموقع *B* فيحدث التالي:

1. تأسيس اتصال (أو ارتباط *Connection*) بين بروتوكولي *(TCP)* وفقاً لطريقة المصافحة الثلاثية *(3-Way Handshaking)*.

2. تبادل المعطيات في كلا الاتجاهين، حيث يزود بروتوكول *(TCP)* خدمة الاتصال المزدوج *(Full Duplex)* والتي يمكن للمعطيات من خلاله أن تتدفق في كلا الاتجاهين في الوقت نفسه. أي إنَّ كل بروتوكول لديه صوان *(Buffer)* للإرسال وآخر للاستقبال، والمقاطع تتحرك في الاتجاهين معاً.

3. إنهاء الاتصال من قبل أحد الطرفين المتصلين (زبون أو مخدم) وغالباً ما يبدأ به الزبون. تسمح معظم التنجيزات بخيارين لإنهاء الاتصال (طريقة المصافحة الثلاثية، وطريقة المصافحة الرباعية مع خيار نصف الإغلاق).

هذا الاتصال هو اتصال افتراضي وليس فيزيائي وتغلف المقاطع في رسالة معطيات *IP*، ويمكن أن ترسل بترتيب مختلف أو أن تفقد أو تفسد وبعد ذلك يعاد إرسالها. قد يسلك كل مقطع مساراً مختلفاً للوصول إلى الوجهة. إذاً يؤسس البروتوكول *TCP* بيئة ذات تدفق موجه يتم فيها الموافقة على مسؤولية تسليم البايتات بالترتيب إلى الموقع الآخر. يظهر الشكل *(7-2)* بنية مقطع *(TCP)* والتي يتم تغليفها من قبل طرد *(IP)* في طبقة الشبكة من بروتوكول *(TCP/IP)*:

Header	Data

Source port address 16 bits								Destination port address 16 bits
Sequence number 32 bits								
Acknowledgment number 32 bits								
HLEN 4 bits	Reserved 6 bits	URG	ACK	PSH	RST	SYN	FIN	Window size 16 bits
Checksum 16 bits								Urgent pointer 16 bits
Options and Padding								

الشكل *7-2*: بنية مقطع ال *TCP*

يتألف المقطع من ترويسة ذات حجم متغير يتراوح بين 20 بايت وبين 60 بايت، تتبع بمعطيات من برنامج التطبيقات. **وفيما يلي نوضح حقول ترويسة المقطع:**

- **رقم البوابة المصدر** ***(Source Port Number)***: يشبه تماماً حقل رقم البوابة المصدر في ترويسة رسالة معطيات *(UDP)*.
- **رقم البوابة الوجهة** ***(Destination Port Number)***: يشبه تماماً حقل رقم البوابة الوجهة في ترويسة رسالة معطيات الـ *(UDP)*.
- **رقم التسلسل** ***(Sequence Number)***: حقل مكون من 32 بت، يحدد الرقم المسند إلى أول بايت معطيات محتوى في هذا المقطع. يعلم رقم التسلسل الوجهة على البايت الأول في التسلسل في المقطع. يستخدم كل اتصال، خلال عملية التأسيس، مولد أرقام عشوائية لإنشاء رقم تسلسلي ابتدائي، والذي يكون مختلف في كل اتجاه.
- **رقم إقرار الاستلام** ***(Acknowledgment Number)***: حقل مؤلف من 32 بت، يحدد رقم البايت المتوقع استقباله من قبل مستقبل المقطع. يمكن إرسال المعطيات والإقرارات معاً وفقاً لطريقة *(Piggybacking)*.
- **طول الترويسة** ***(Header Length)***: حقل مؤلف من 4 بتات، يشير إلى عدد الكلمات (الكلمة مرمزة على 4 بايت) في ترويسة *TCP*. طول الترويسة يمكن أن يكون ضمن المجال *[20 - 60]* بايت. لذلك فإنَّ قيمة هذا الحقل هي رقم ضمن المجال *[5 - 15]*، لأنَّ الكلمة على 4 بايت.
- **محجوز** ***(Reserved)***: حقل مؤلف من 6 بتات مخصص للاستخدامات المستقبلية.

• **التحكم (*Control*)**: يُعرِّف هذا الحقل *6* بتات تحكم مختلفة أو رايات (*flags*) كما هو موضح في الشكل (*2-8*). تُفَعِّل هذه البتات التحكم بالتدفق وتأسيس وإنهاء الاتصال وإغلاق الاتصال وتحدد نمط نقل المعطيات.

URG	ACK	PSH	RST	SYN	FIN

الشكل *2-8*: حقول التحكم في ترويسة ال *TCP*

ويبين الجدول التالي معنى ووظيفة كل بت في حقل التحكم:

الراية (*Flag*)	الوصف (*Description*)
URG	**قيمة حقل مؤشر الطوارئ تكون صحيحة**
ACK	**قيمة حقل رمز إشعار الاستلام تكون صحيحة**
PSH	**دفع البيانات**
RST	**إعادة الاتصال إلى حالتها الأولى**
SYN	**أرقام تسلسل التزامن خلال إنشاء الاتصال**
FIN	**إنهاء الاتصال**

الجدول *2-1*: وصف بتات التحكم

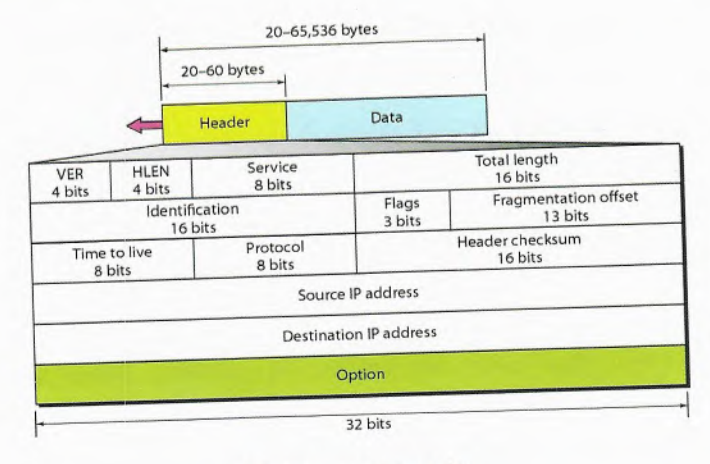
• **حجم النافذة (*Window Size*)**: يُعَرِّف هذا الحقل حجم نافذة الاستقبال بالبايتات. وهو مكون من *16* بت. أي إنَّ الحجم الأعظمي للنافذة هو *65,535* بايت.

- **اختبار المجموع (*Checksum*):** حقل مؤلف من *16* بت. تتبع إجرائية حسابه إجرائية اختبار المجموع المستخدمة في *UDP* نفسها.
- **مؤشر الاستعجال (*Urgent pointer*):** حقل مؤلف من *16* بت، يكون ساري المفعول فقط في حال كانت راية *(URG)* مفعّلة وهو يستخدم في حال وجود بيانات مستعجلة. هذا الحقل يُعَرِّف الرقم الذي يجب أن يضاف إلى رقم التسلسل للحصول على رقم آخر بايت مستعجل في جزء المعطيات من المقطع.
- **خيارات (*Options*):** يمكن أن يصل هذا الحقل إلى *40* بايت من المعلومات الاختيارية في ترويسة *TCP*. يستخدم عادةً هذا الحقل لحساب تأخير وصول الطرود ***(RTT)*** ولمضاعفة حجم نافذة المستقبل وللتفاوض على الطول الأعظمي للنقل *Maximum* (*Transfer Unit - **MTU***) بين الطرفين.

2 - 8 - البروتوكول *(IP)*

يعرف هذا البروتوكول ببروتوكول الإنترنت *(Internetwork Protocol)*. يقوم هذا البروتوكول بإرسال واستقبال الطرود وعملية التوجيه لها من المرسل إلى المستقبل وذلك حسب عنوان محدد للوجهتين.

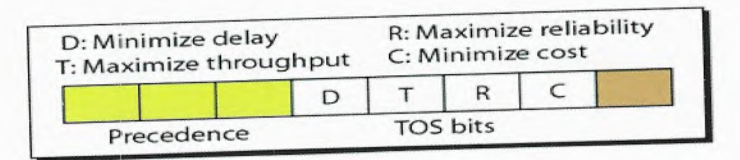
بالإضافة لإمكانيته القيام بعملية تجزئة الطرود وإعادة تجميعها، يقوم هذا البروتوكول بتحقيق واجهة ربط بين مضيفين وذلك بسبب إمكانيات التجزئة والتجميع، لذلك فهو يؤمن للطبقة الأعلى إمكانية استخدام قنوات ربط بين المضيفين. يبين الشكل *(2-10)* المخطط العام لشكل طرد *IPv4*:



الشكل 2-10: ترويسة طرد IP

حقول الترويسة

- **الحقل (VER):** يمثل رقم إصدار البروتوكول وفي حالتنا هو 4.
- **الحقل (HLEN):** يمثل طول الترويسة بعد ضرب ب الرقم 4 وهو يتراوح بين 5 و15، أي أن طول الترويسة يتحدد بين 20 و60.
- **الحقل نوع الخدمة (Type Of Service - TOS):** يتكون من عدة بتات يتم تمثيله على الشكل التالي:



الشكل 2-11: حقل نوع الخدمة

حيث يمثل D تقليل التأخير، وT زيادة معدل النقل، وR زيادة الموثوقية، وC تقليل التكلفة.

- **الحقل** *(Total Length)*: يمثل طول المعطيات مع الترويسة في الطرد.
- **الحقل** *(Flags)*: يتألف من ثلاثة بتات، الأول غير مستخدم، والثاني هو ***D*** يعني عدم التجزئة، والثالث ***M*** يمثل أن هنالك أجزاء أخرى من الطرد الأصلي غير هذا الجزء.
- **الحقل** *(Fragment Offset)*: يمثل قيمة انزياح المعلومات في الطرد عن بداية الرسالة وذلك بعد ضرب محتوى الطرد بـ *8*.
- **الحقل** *(TTL)*: يحدد عدد الموجهات *(Routers)* أو الأجهزة التي تحوي البروتوكول*IP* التي يمكن للطرد أن يعبرها وهو مفيد جداً في حالات وجود الحلقات.
- **الحقل** *(Protocol)*: يحدد هذا الحقل البروتوكول الذي يأتي مباشرة فوق *IP* ويمثل الجدول التالي هذه القيم:

Value	Protocol
1	ICMP
2	IGMP
6	TCP
17	UDP
89	OSPF

الجدول *2-2*: **قيم الحقل بروتوكول** *(Protocol)*.



قيم الحقل (*Protocol*) في طرد *IP*

- **الحقل (*Headers Checksum*):** تستخدم للتحقق من خلو ترويسة الطرد المستلم من الأخطاء.
- **الحقل (*Source Address*):** يحدد عنوان المرسل أو (*IP* المرسل) وهو يتألف من*4* بايتات.
- **الحقل (*Destination Address*):** يحدد (*IP* المستقبل) وهو يتألف من *4* بايتات.

عناوين *IPv4*

يتكون عنوان *IP* من *32 bits* وهو يُعرَف تعريفاً وحيداً على مستوى العالم بأسره. نقصد بكون عنوان *IP* وحيداً أنه يعرف اتصالاً وحيداً بالإنترنت. لا يمكن لجهازين واقعين على الإنترنت امتلاك نفس عنوان *IP* في نفس الوقت. ويمكن لجهاز امتلاك عنوان *IP* لفترة محددة من الوقت، كفترة اتصاله بالإنترنت، ومن ثم تحرير العنوان لجهاز آخر. إذا احتوى جهاز ما على عدة وصلات مع الإنترنت فيجب تخصيص عنوان *IP* خاص لكل وصلة.

فضاء العنونة

يعرّف فضاء العنونة بأنه المجموع الكلي للعناوين التي يستخدمها بروتوكول *IPv4*. فإذا استخدم بروتوكولاً ما *N bits* لتعريف العنوان فإن فضاء العنونة يكون 2^N.

يستخدم الإصدار الرابع من بروتوكول *IP 32* بت للعنونة مما يعني أننا نستطيع عنونة *232* جهازاً أو *4,294,967,296*. لكن لم يتم استخدام فضاء العنونة هذا بشكل جيد بسبب بعض القيود الموضوعة عليه الأمر الذي أدى إلى هدر جزء كبير من العناوين.



التدوين *Notation*

يمكن تدوين عنوان *IP* باستخدام طريقتين:

1. التدوين الثنائي *(Binary)*:

يجري هنا كتابة العنوان على شكل *32* بت *(أو أربعة بايتات)* مثل العنوان التالي:

01110101 10010101 00011101 00000010

2. التدوين العشري المنقط *(Dotted-Decimal Notation)*:

يفيد التدوين العشري في جعل العنوان أقل حجماً وأسهل للقراءة والحفظ. نكتب هنا كل بايت من البايتات الأربعة المكونة للعنوان بالشكل العشري مع إضافة نقطة بين البايتات مثل العنوان التالي: *117.149.29.2*

العنونة المصنفة *(Classful Addressing)*

توزع العناوين في هذا النوع إلى عدة أصناف، ويكون عنوان *IP* موزعاً على خمسة أنواع هي: *A, B, C, D, and E*. يشغل كل صنف جزءاً من فضاء العنونة المتاح.

يمكننا معرفة الصنف الذي ينتمي إليه عنوان ما من خلال الخانات الثنائية الأولى للعنوان *(من اليسار)* إذا كان مدوناً بالصيغة الثنائية، أو من خلال البايت الأول من اليسار إذا كان مدوناً بالصيغة العشرية.

يبين الشكل التالي *(2-12)* تصنيفات العنونة المستخدمة وبتات أو بايتات الدالّة على نوع الصنف.

	First byte	Second byte	Third byte	Fourth byte
Class A	0			
Class B	10			
Class C	110			
Class D	1110			
Class E	1111			

a. Binary notation

	First byte	Second byte	Third byte	Fourth byte
Class A	0–127			
Class B	128–191			
Class C	192–223			
Class D	224–239			
Class E	240–255			

b. Dotted-decimal notation

الشكل *2-12*: إيجاد صفوف العناوين وفق التدوين الثنائي أو العشري

إيجاد تصنيف العناوين وفق التدوين الثنائي أو العشري

تكمن المشكلة مع هذا النوع من العناوين في كونها مقسمة إلى أعداد ثابتة من الكتل وكل كتلة مؤلفة من حجم ثابت أيضاً.

الصف	عدد الكتل	حجم الكتلة	الاستخدام
A	*128*	*128*	وحيد الوجهة *Unicast*
B	*65,536*	*16,384*	وحيد الوجهة
C	*256*	*2,097,152*	وحيد الوجهة
D	*268,435,456*	*1*	متعدد الوجهات *Multicast*
E	*268,435,456*	*1*	محجوز

الجدول *2-3*: عدد الكتل وحجمها ضمن صفوف عناوين *IPv4*

الصنف D مخصص للإرسال متعدد الوجهات والصنف E مخصص للتجارب.



الصنف *A* مثالاً عن العناوين المصنفة

يستخدم البت الأول والي يكون دائما صفر لتحديد ان هذا العنوان هو من الصنف *A* والبتات السبعة الباقية من البايت الأول تستخدم لتحديد الشبكة.

البتات الثلاث التالية تستخدم من أجل تحديد عنوان المضيف في هذا الصنف. ويحوي هذا الصنع على *128* عنوان للشبكة بالإجمال، ولكن العناوين التي تكون مؤلفة كافة خاناتها من أصفار لا تستخدم، والعنوان *127* هو عنوان خاص له استخدامات خاصة أيضا. لذلك فعناوين الشبكة المتاحة في هذا الصنف هي *126* عنوان متاح للشبكة.

ويكون عدد عناوين المضيفين المتاحة في هذا الصنف هو:

$(2^n - 2) = 16,777,214$، حيث أن n تمثل عدد الخانات في قسم المضيف. وهي في هذا الصنف تساوي *24*.

معرف الشبكة *(Netid)* ومعرف المضيف *(Hostid)*

يجري تقسيم عنوان *IP* في الأصناف فقط *A, B, C* إلى معرف شبكة ومعرف مضيف ذات أطوال متغيرة ومتعلقة بصف العناوين.

يخصص العنوان *255.255.255.255* كعنوان البث، وعندما يرسل مضيف ما رسالة تتضمن هذا العنوان كعنوان هدف، يتم توصيل الرسالة إلى كافة المضيفين المتواجدين في الشبكة الفرعية نفسها.

يبين الجدول التالي بعض البايتات المخصصة لمعرف الشبكة باللون الأزرق ولمعرف المضيف.

الصنف	التمثيل الثنائي لقناع الشبكة	التمثيل العشري لقناع الشبكة	CIDR
A	*11111111 0000000 00000000 00000000*	*255.0.0.0*	*/8*
B	*11111111 11111111 00000000 00000000*	*255.255.0.0*	*/16*
C	*11111111 11111111 11111111 00000000*	*255.255.255.0*	*/24*

الجدول *4-2*: قناع الشبكة الافتراضي للعناوين ذات التصنيف

قناع الشبكة الافتراضي للعناوين المصنفة *(Classful Addresses)*

يستخدم ضمن الصنف *A* بايتاً واحداً للدلالة على معرف الشبكة وثلاثة بايتات للدلالة على معرف المضيف (المحطة) داخل الشبكة. بينما يستخدم، ضمن الصف *B*، بايتين للدلالة على معرف الشبكة وبايتين للدلالة على معرف المضيف. ونخصص أخيراً ثلاثة بايتات لمعرف الشبكة وبايت واحد لمعرف المضيف ضمن الصنف *C*.

مفهوم قناع الشبكة *Net Mask*

على الرغم من أن الأقسام المخصصة لمعرف الشبكة وتلك المخصصة لمعرف المضيف محددة تحديداً ثابتاً، إلا أننا نستطيع أن نستخدم مفهوم قناع الشبكة الذي يتكون من *32* بت. يبين الجدول *4-2* قناع الشبكة الافتراضي المستخدم لأصناف العناوين*A, B, C* فقط.

يفيد القناع في إيجاد الجزء المخصص لمعرف الشبكة أو لمعرف المضيف من عنوان *IP* ما. فمثلاً بما أن قناع الشبكة للصف *A* مكون من ثمانية واحدات متتالية فهذا يعني أن أول بايت من العنوان هو عنوان الشبكة



(أي أن الواحدات المتتالية تحدد معرف الشبكة والأصفار المتتالية تحدد معرف المضيف).

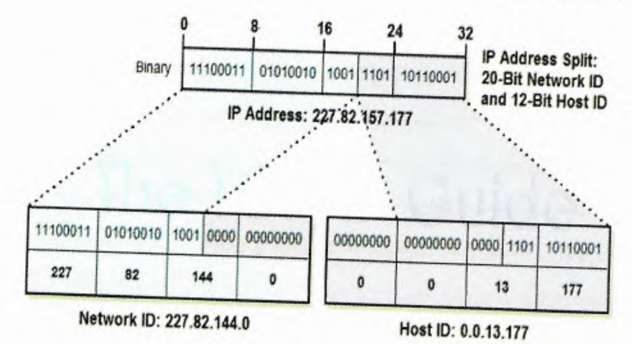
يظهر العمود الأخير من الجدول القناع مكتوباً بالصيغة /*n* حيث يأخذ المتحول *n* القيم *(8، 16، أو 24)*. يدعى هذا التدوين *(Slash Notation)* أو *(Classless InterDomain Routing – CIDR)*. ويستخدم هذا التدوين في عناوين *IP* غير المصنفة.

تستخدم عملية تطبيق قناع الشبكة الفرعية عدة عمليات من الجبر البولياني لتصفية وعدم تضمين الخانات التي لا تتطابق في تعريف عنوان الشبكة. إن التدوين *CIDER* يعتمد في هذا المجال على الطول المتغير لقناع الشبكة الفرعية *(variable - length subnet masking (VLSM)*.

مثال عن تحديد معرف الشبكة ومعرف المضيف

لنفرض لدينا عنوان انترنت يعرّف فيه *20* خانة لعنوان الشبكة والباقي لتعريف عنوان المضيف.

يظهر الشكل أدناه كيفية تقسيم ومن ثم حساب العناوين لكل من الشبكة والمضيف.





2 - 9 - مبادئ التطبيقات الشبكية

سنستخدم من الآن فصاعداً مصطلح نظام نهائي *ES* أو مضيف للدلالة على أي حساب أو مخدم أو هاتف ذكي قادر على تشغيل أي تطبيق شبكي

تعتبر كتابة برامج لتطوير تطبيقات شبكية تعمل على أنظمة نهائية *(ES–End Systems)* مختلفة وتتخاطب مع بعضها البعض عبر الشبكة هي أساس التطبيقات الشبكية. فضمن تطبيق الوب، على سبيل المثال، يوجد برنامجان مختلفان يتخاطبان مع بعضهما: متصفح الإنترنت الذي يعمل على حاسب المستثمر (سنطلق عليه اسم المضيف *Host* لأنه يمكن أن يكون حاسب شخصي أو محمول أو هاتف ذكي أو هاتف لوحي) وبرنامج مخدم الوب الذي يعمل على مضيف مخدم الوب. بينما يوجد، في شبكات مشاركة الملفات من نوع *P2P*، برنامج ضمن كل مضيف يشارك في عملية نقل الملفات أي تكون البرامج ضمن المضيفين هنا متماثلة.

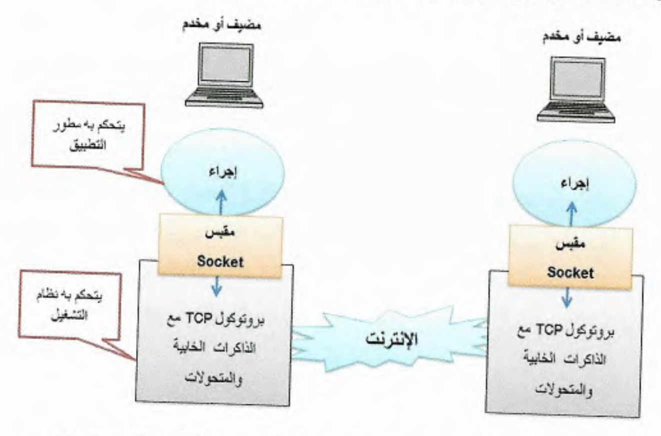
2-4-1 بنيان التطبيقات الشبكية *Network Applications Architecture*

يختلف بنيان التطبيق الشبكي عن بنيان الشبكة نفسها الذي يتكون من مجموعة من الطبقات تزود كل واحدة مجموعة من الخدمات للطبقة الأعلى منها مباشرةً. بينما يقوم مصمم التطبيق بتصميم بنيان التطبيق الذي يعمل بين مجموعة من الأنظمة النهائية. هنا يحتاج المصمم إلى الاختيار بين أحد البنيانين أو لنموذجين المعروفين ضمن التطبيقات الشبكية والذين تم تعريفهما في الفصل الأول وهما مخدم-الزبون والند للند *P2P*.



واجهة التخاطب بين الإجراء والشبكة

يجري تبادل الرسائل بين إجراءي المخدم والزبون عبر الشبكة باستخدام واجهة تخاطب برمجية تدعى المقبس *(Socket)*. يبين الشكل التالي آلية التخاطب بين الإجراءات باستخدام بروتوكول *TCP*.



الشكل *2-13*: التخاطب بين الإجراءات عن طريق الإنترنت باستخدام المقابس

كما هو موضح بالشكل السابق، المقبس هو صلة الاتصال بين التطبيق وبين طبقة النقل ضمن المضيف. يطلق عليه أيضاً اسم واجهة برمجة التطبيقات *Application Programming Interface (API)*. يملك مطور التطبيقات تحكماً كاملاً من طرف طبقة التطبيقات للمقبس بينما يملك تحكماً ضئيلاً من طرف طبقة النقل للمقبس. التحكم الوحيد الذي يستطيع المطور فرضه على طرف طبقة النقل للمقبس هو: *(1)* اختيار طبقة النقل و*(2)* تثبيت بعض وسطاء النقل مثل حجم الصوان *(Buffer size)*، وطول مقطع الشبكة الأعظمي *(Maximum MTU)*. بعد اختيار طبقة النقل، يستطيع المطور بناء التطبيق باستخدام خدمات طبقة النقل التي جرى اختيارها.



عنونة الإجراءات *Addressing Processes*

حتى يستطيع إجراء يعمل ضمن مضيف ما التخاطب مع إجراء آخر يعمل ضمن مضيف آخر، يجب على الإجراء الوجهة أن يملك عنواناً. يتألف العنوان من جزأين: *(1)* عنوان للمضيف و*(2)* معرف يحدد الإجراء الوجهة ضمن المضيف الوجهة.

يجري تعريف المضيف، ضمن الإنترنت، باستخدام عنوان *IPv4* المكون من *4* بايتات أو *IPv6* المكون من *16* بايتاً بينما يجري تعريف الإجراء (أو المقبس الوجهة) ضمن المضيف باستخدام رقم البوابة *(Port Number)* الممتد على *2* بايت. نحتاج إلى تعريف رقم البوابة للمقبس لأنه يمكن لمضيف ما أن يحوي عدة تطبيقات شبكية في الوقت نفسه. جرى تخصيص التطبيقات المعروفة برقم بوابات معروفة أيضاً. فخصص مثلاً مخدم الويب بالبوابة *80* وخصص مخدم البريد الالكتروني بالبوابة *25*.

خدمات النقل المتوافرة للتطبيقات

نذكّر هنا أن التطبيق الذي يعمل عند المرسل يدفع الرسائل عبر المقبس حيث تقوم طبقة النقل عنده بنقل الرسائل لمقبس الإجراء الوجهة.

توفر الإنترنت خيارين للخدمات التي يمكن أن تقدمها طبقة النقل، وهما: *TCP* أو *UDP*. يقدم كل منهما مجموعة من الخدمات إلى التطبيق.



خدمات *TCP*

عند اختيار طبقة النقل *TCP* ليعمل فوقها التطبيق فإننا نتوقع الحصول على الخدمات التالية:

- خدمه اتصال موجهة *(Connection-oriented service)*.
- خدمة نقل معطيات موثوقة بدون أخطاء.
- خدمة التحكم بالتدفق وبالاختناقات.

خدمات *UDP*

تقدم طبقة *UDP* مجموعة ضئيلة من الخدمات فهي عديمة التوجيه *(Connectionless)* وغير موثوقة، ولا تضمن ترتيب الطرود المستقبلة. لا تعالج *UDP* الاختناقات مع الشبكة ولا تتحكم بتدفق الطرود مع الوجهة.

2 - 10 - تمارين محلولة

1. طبقة التقديم في النموذج المعياري للشبكات *(OSI)* هي:

a. الطبقة السابعة

b. الطبقة الخامسة

***c*. الطبقة السادسة**

d. الطبقة الرابعة

e. الطبقة الثانية.

2. ترمز *(FTP)* إلى :

***a*. File Transfer Protocol**

b. *File Transmission Protocol*

c. *Form Transfer Protocol*

d. *Form Transmission Protocol*

e. *Firewire Transfer Protocol*

3. البروتوكول هو:

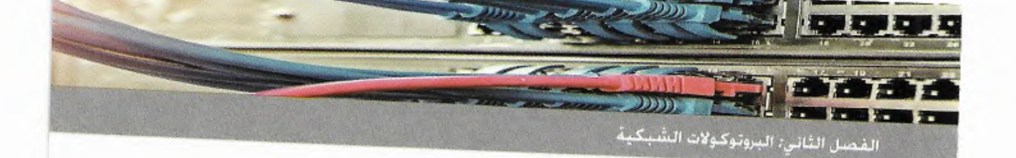
a. مجموعة من الآليات المستخدمة لتنفيذ مهمات معينة

b. تعريف التقانات المستخدمة

***c*. مجموعة من القواعد والاتفاقات التي تحكم تبادل البيانات بين طرفين**

d. كل ما سبق

e. ولا إجابة مما سبق.



4. من العناصر الرئيسية في تعريف البروتوكولات الشبكية:

a. *Entity*

b. *Host*

c. *Syntax*

d. *Semantic*

e. **كل من (c) و(d).**

5. من ضمن وظائف البروتوكولات الشبكية:

a. *Encapsulation*

b. *Segmentation*

c. *Synchronization*

d. **كل ما سبق**

e. كل من (a) و(b).

6. طبقة التطبيقات في النموذج المعياري للشبكات *OSI* هي:

a. **الطبقة السابعة**

b. الطبقة الخامسة

c. الطبقة السادسة

d. الطبقة الرابعة

e. الطبقة الأولى.

7. تستخدم شبكة *(FDDI)* الطبولوجيا الفيزيائية التالية:

a. *Bus*

b. ***<u>Ring</u>***

c. *Star*

d. *Tree*

e. *Mesh*

8. ماذا تسمى الإشارة التي تنتقل حول *(token-ring)* وتحمل المعطيات؟

a. *Packet*

b. *Frame*

c. ***<u>Token</u>***

d. *Bus*

e. ولا إجابة مما سبق.

9. أي من طبقات *OSI* تتعامل مع العنوان الفيزيائي للجهاز؟

a. طبقة الإيثرنت

b. طبقة *IP*

c. الطبقة الفيزيائية

d. **<u>طبقة ترابط المعطيات *(Data link)*</u>**

e. ولا إجابة مما سبق.



10. في شبكات *IP* تفصل عناوين الشبكة عن المضيفين باستخدام:

ARP .a

TCP .b

Gateway .c

<u>*Netmask*</u> .**d**

ICMP .e

11. ما هو عنوان الانترنت للبث *(broadcast)*؟

a. هو عنوان الانترنت مع كل الخانات الثنائية للمضيف مضبوطة على الصفر

b. هو عنوان الإنترنت مع كل الخانات الثنائية للشبكة مضبوطة على *1*

c. **<u>هو عنوان الانترنت مع كل الخانات الثنائية للمضيف مضبوطة على *1*</u>**

d. هو عنوان الانترنت الذي يكون فيه البايت الأخير مضبوط على *255*.

e. ولا إجابة مما سبق.

12. ترمز *PDU* إلى:

Personal Data Unit .a

<u>Protocol Data Unit</u> .**b**

Protocol Data Universal .c

Protocol Data Unigram .d

.e ولا إجابة مما سبق.



13. التحكم بالتدفق:

a. هو اتصال موجه بين طرفين.

b. هو تسليم مرتب للطرود

c. هو عملية تحديد معدل تدفق البيانات من قبل الطرفية المرسلة

d. **هو عملية تحديد معدل تدفق البيانات من قبل الطرفية المستقبلة**

e. ولا إجابة مما سبق.

14. النموذج الطبقي المرجعي *(OSI)* هو:

a. أكثر البروتوكولات استخدماً في الشبكات

b. النموذج المرجعي الأكثر استخداماً تجارياً

c. **نموذج لفهم وتصميم الشبكات المعتمد عالمياً**

d. كل ما سبق

e. ولا إجابة مما سبق.

15. من مهام الطبقة الفيزيائية:

a. التحكم بالأخطاء

b. التحكم بالتدفق

c. **تحديد طبولوجية الشبكة**

d. العنونة الفيزيائية

e. التحكم بالتأخير

16. الطبقة التي تقوم بإضافة كل من ترويسة ولاحقة من النموذج *(OSI)* هي:

a. *Physical*

b. ***<u>Data Link</u>***

c. *Network*

d. *Transport*

e. *Session*

17. تتألف الطبقة ------- من طبقتين جزئيتين هما *(LLC)* و*(MAC)* :

a. *Transport*

b. *Session*

c. *Physical*

d. *Presentation*

e. ***<u>Data Link</u>***

18. مهمة البروتوكول *(ARP)* في الشبكة هي:

a. ترجمة العناوين الفيزيائية للشبكة إلى عناوين منطقية.

b. ترجمة العنوان *IP* إلى عنوان منطقي.

c. ترجمة عنوان فيزيائي إلى عنوان *IP*.

d. **<u>ترجمة عنوان *IP* إلى عنوان فيزيائي.</u>**

e. ولا إجابة مما سبق.



19. **يتألف عنوان الإجراء من ________ :**

a. ***IP address and port number***

b. *TCP address and IP address*

c. *MAC address and IP address*

d. *Port number and Domain name*

e. *Logical Address*

20. **يوجد حالياً خدمتي نقل أساسيتين متوفرتين للتطبيقات، هما ________ :**

a. *TCP and HTTP*

b. *TCP and Ethernet*

c. ***TCP and UDP***

d. *UTP and STP*

e. ولا إجابة مما سبق.



الفصل الثالث

مكونات الشبكة

تختلف مكونات الشبكة حسب نوع الشبكة المطلوب تركيبها. فالشبكات المحلية، على سبيل المثال، تتطلب تركيب بطاقة شبكة *NIC (Network Interface card)* على كل جهاز وتمديد كابلات نحاسية إلى عقدة مركزية تدعى المبدلة *Switch*.

3 - 1 - بطاقة الشبكة *NIC*

بطاقة الشبكة أو محول الشبكة *Network adapter* هو عبارة عن دارة الكترونية يجري تثبيتها ضمن الحاسوب حتى يستطيع الاتصال بالشبكة.



الشكل *1*- بطاقة الشبكة

تعتبر بطاقة الشبكة الواجهة التي تصل بين جهاز الحاسوب وكابل الشبكة وبدونها لا تستطيع الحواسيب الاتصال فيما بينها من خلال الشبكة.

3 - 1 - 1 - وظائف بطاقة الشبكة

تقوم بطاقة الشبكة بالوظائف التالية:

- تحضير البيانات لبثها على الشبكة.
- إرسال البيانات على الشبكة.
- التحكم بتدفق البيانات بين الحاسوب ووسط الإرسال.
- تحويل الإشارات الكهربية من كابل الشبكة إلى بتات يفهمها معالج الحاسوب، وعندما تريد إرسال بيانات فإنها تترجم إشارات الحاسوب الثنائية إلى إشارات كهربية يستطيع سلك الشبكة حملها.

تمتلك كل بطاقة شبكة عنوان فيزيائي مكون من 6 بايتات تكتب كالمثال التالي:

06 : 01 : 02 : 01 : 2C : 4B

لاحظ أن كل جزء من العنوان السابق يمثل بايتاً واحداً مكتوباً بالترميز الست عشري.

3 - 2 - وسائط النقل

تقسم وسائط النقل إلى نوعين: موجهة *guided* وغير موجهة *unguided*.

3 - 2 - 1 - وسائط النقل الموجهة

تقع وسائط النقل تحت الطبقة الفيزيائية التي تتحكم بها تحكماً مباشراً.



تعريف: وسيط النقل بشكل عام هو أي شيء قادر على حمل البيانات من مصدر إلى وجهة.

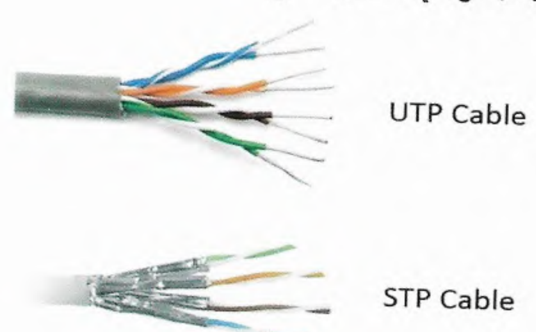
أما في مجال الشبكات الحاسوبية، فيصبح وسيط النقل هو الفضاء الخارجي أو الكابلات المعدنية أو كابلات الألياف الضوئية.

نبين فيما يلي الأنواع المختلفة لوسائط النقل الموجهة:

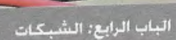
1 - كابلات الأزواج المجدولة *Twisted-pair cable*

تنقل الأزواج المجدولة الإشارات على شكل تيار كهربائي. يتألف كل زوج مجدول من ناقلين نحاسيين، يغطي كل منهما عازل بلاستيكي ويجدلان مع بعضهما. يفيد الجدل في زيادة مقاومة الكابل للضجيج الخارجي وتقليل التداخلات الكهرطيسية بين الأزواج والتي تعرف بالتسميع *Crosstalk*.

يوجد نوعين من كابلات الأزواج المجدولة: غير المحمية *Unshielded Twisted Pair (UTP)* والمحمية *Shielded Twisted Pair (STP)* كما هو موضح في الشكل التالي.



الشكل -2 أنواع كابلات الأزواج المجدولة



لاحظ أن الكابلات المحمية تحوي حماية معدنية لكل زوج أسلاك معزول.

تفيد الحماية في زيادة مقاومة الكابل ضد الضجيج الخارجي والتداخلات البينية لكنه يصبح أكبر حجماً وأغلى ثمناً.

عرفت المؤسسة الصناعية للإلكترونيات *Electronic Industries Association (EIA)* معايير لكابلات الأزواج المجدولة مصنفة وفق سبعة أصناف مختلفة.

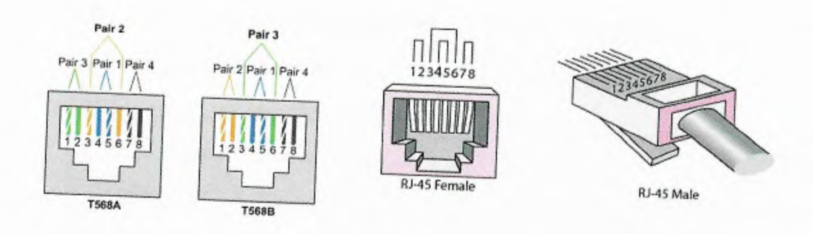
UTP Category	Data Rate	Max. Length	Cable Type	Application
CAT1	Up to 1Mbps	-	Twisted Pair	Old Telephone Cable
CAT2	Up to 4Mbps	-	Twisted Pair	Token Ring Networks
CAT3	Up to 10Mbps	100m	Twisted Pair	Token Rink & 10BASE-T Ethernet
CAT4	Up to 16Mbps	100m	Twisted Pair	Token Ring Networks
CAT5	Up to 100Mbps	100m	Twisted Pair	Ethernet, FastEthernet, Token Ring
CAT5e	Up to 1 Gbps	100m	Twisted Pair	Ethernet, FastEthernet, Gigabit Ethernet
CAT6	Up to 10Gbps	100m	Twisted Pair	GigabitEthernet, 10G Ethernet (55 meters)
CAT6a	Up to 10Gbps	100m	Twisted Pair	GigabitEthernet, 10G Ethernet (55 meters)
CAT7	Up to 10Gbps	100m	Twisted Pair	GigabitEthernet, 10G Ethernet (100 meters)

الشكل -3 أصناف الكابلات المجدولة

يعتبر موصل *RJ45* من أشهر الموصلات التي تستخدمها كابلات الأزواج المجدولة.

إذا أخذنا كابل شبكة إيثرنت المحلية على سبيل المثال، فنشاهد أنه يتألف من ثمانية ألوان. يجري جدل كل لونين معاً لتشكيل زوج.

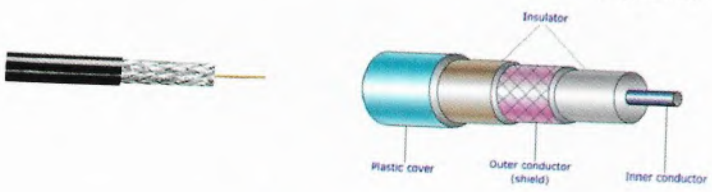




الشكل -4 موصلات *RJ45*

2 - الكابلات المحورية

الكابلات المحورية هي كابلات كهربائية أنبوبية الطبقات. تجري حماية الكابل الكلي بغلاف بلاستيكي يغطي بدوره طبقة عزل الناقل. يفيد الناقل الخارجي في حماية الكابل من الضجيج كما يعمل كناقل ثانٍ لتشكيل الدارة الكهربائية. يفصل الناقلين عازل آخر كما هو مبين في الشكل *5*.



الشكل -5 الكابل المحوري

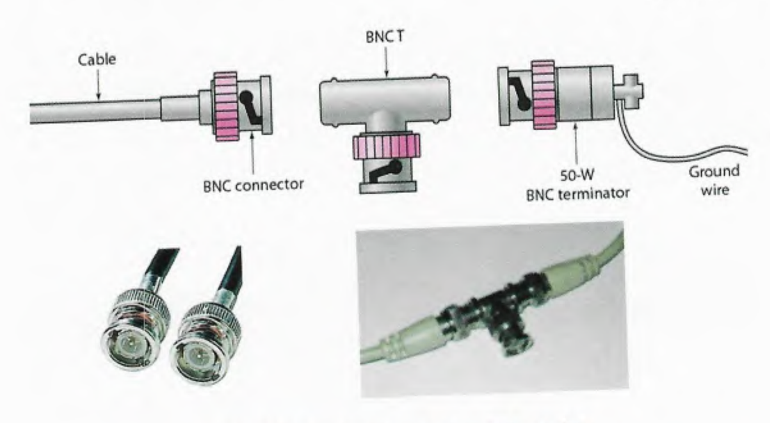
نحتاج إلى موصلات للكابل المحوري لوصل الكابل مع الجهاز. يعتبر موصل *Bayone-Neil-Concelman (BNC)* من أشهر الموصلات المستخدمة مع الكابلات المحورية.

الأنواع الثلاثة المعروفة للموصلات هم: موصل *BNC* المستخدم لوصل نهاية الكابل مع جهاز ما؛ وموصل نهاية *BNC* المستخدم في نهاية

الكابل لامتصاص الإشارات ومنع ارتدادها؛ وموصل *BNC-T* المستخدم في شبكات إيثرنت لتوصيل بطاقة شبكة إلى كابل محوري.

لقد جرى استخدام الكابلات المحورية سابقاً في شبكات الهاتف التماثلية *Analog* وشبكات إيثرنت المرية *Bus topologies*. أما في وقتنا الحالي، فقد تم استبدالها بكابلات الأزواج المجدولة وكابلات الألياف الضوئية.

يبين الشكل *6* أنواع الموصلات المستخدمة مع الكابل المحوري وطريقة الوصل.



الشكل -6 الموصلات المستخدمة مع الكابل المحوري

3 - كابلات الألياف الضوئية *Fiber-Optic cables*

تصنع كابلات الألياف الضوئية من الزجاج أو البلاستيك وتنقل الإشارات على شكل ضوء. يكون الليف الضوئي رقيقاً ومرناً وشفافاً يعمل على توجيه الضوء ضمنه الأمر الذي يسمح له بنقل الإشارة الضوئية بين نهايتي الليف. تسمح الألياف الضوئية بنقل الإشارات إلى مسافات أطول وبمعدلات نقل أكبر من الأنواع الأخرى.

يستخدم الليف الضوئي ظاهرة الانكسار *Reflection* لتوجيه الضوء ضمن الليف. يجري هنا استخدام نواة داخلية مصنعة من الزجاج أو البلاستيك تدعى *core*، تحيط به كسوة *Cladding* مصنوعة من الزجاج أو البلاستيك الأقل كثافةً. يسمح فرق الكثافة بين النواة والكسوة بجعل الضوء ينعكس عن طبقة الكسوة ويبقى ضمن النواة.

تعتبر كابلات الألياف الضوئية محصنة ضد التداخلات الكهرطيسية نظراً لعدم وجود المعادن ضمنها.

3 - 1 - 2 - وسائط النقل غير الموجهة

وسائط النقل غير الموجهة هي وسائط النقل اللاسلكية حيث لا يوجد أي ناقل فيزيائي بين المصدر والوجهة. يدعى هذا النوع من الاتصالات بالاتصالات اللاسلكية *Wireless communication*.

يجري هنا بث الإشارات عبر الهواء حيث تصبح متوافرة لأي جهاز يمتلك الآليات المناسبة لاستقبالها.

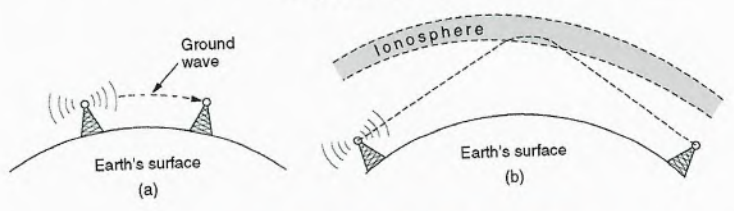
يمكن أن تنتشر الإشارات غير الموجهة بطرق عدة: إما باستخدام الانتشار الأرضي *Ground propagation* أو الانتشار الفضائي *Sky propagation* أو الانتشار وفق خط النظر *Line-of-sight propagation*.

في الانتشار الأرضي، تنتقل الأمواج الراديوية من خلال الجزء الأدنى للغلاف الجوي بالقرب من الأرض. في هذه الحالة، تنتشر الأمواج الراديوية ذات الترددات المنخفضة في جميع الاتجاهات مع ملاحقة انحناء الكرة الأرضية.

في الانتشار الفضائي، تنتشر الأمواج الراديوية ذات الترددات الأعلى بشكل متصاعد حتى تصل إلى طبقة الأيونوسفير حيث تنعكس إلى الأرض مجدداً. يسمح هذا النوع من الانتشار بالوصول إلى مسافات طويلة باستخدام طاقة منخفضة.

في الانتشار وفق خط النظر، تنتشر الأمواج الكهرطيسية وفق خطوط مستقيمة من هوائي لآخر. تكون الهوائيات اتجاهية *Directional* ومواجهة لبعضها البعض ومرتفعة عن سطح الأرض حتى لا تحجب انحناءات الكرة الأرضية خط النظر بين الهوائيين.

يمكن استخدام الأشعة تحت الحمراء للاتصالات القريبة كتلك المستخدمة بين الحاسوب وبعض الطرفيات.



الشكل 7- *(a)* الانتشار الأرضي و*(b)* الانتشار الفضائي

يجري هنا استخدام أمواج الاتصال اللاسلكي لنقل البيانات على مسافات قصيرة أو طويلة. يستخدم المصطلح «لاسلكي *wireless*» للدلالة على الاتصالات التي من خلالها تحمل الأمواج الكهرطيسية الإشارة على كامل المسار أو جزء منه.

يقسم النقل اللاسلكي إلى ثلاث مجموعات. تضم هذه المجموعات الأمواج الراديوية *radiowaves* والأمواج الميكروية *microwaves* وأمواج الأشعة تحت الحمراء *infrared waves*. تُستخدم الأمواج الراديوية للبث مثل البث التلفزيوني أو بث محطات الراديو.

تفيد الأمواج الميكروية في الاتصال بين مرسل ومستقبل. تستخدم أيضاً في الهواتف النقالة وشبكات الأقمار الاصطناعية والشبكات المحلية اللاسلكية *Wireless LANs*.

أما أمواج الأشعة تحت الحمراء، فيجري استخدامها في المسافات القصيرة كوصل لوحة المفاتيح والفأرة والطابعة مع جهاز الحاسوب.

الأمواج الراديوية هي متعددة الاتجاهات *omni-directional* بمعنى أن الإشارة تنتشر في جميع الاتجاهات ولا حاجة لمحاذاة المرسل مع المستقبل وجهاً لوجه لتفعيل استقبال الإشارات.

نطاق الأمواج الراديوية ضيق لأنه يشمل الأمواج ذات التردد الأقل من *GHz 1*.

أدى تقسيم هذا النطاق إلى نطاقات جزئية أصغر إلى الوصول إلى اتصالات رقمية ذات معدل نقل معطيات قليل.

أما الأمواج الكهرطيسية ذات التردد المحصور بين *1GHz* و *300GHz* فتدعى الأمواج الميكروية. الأمواج الميكروية وحيدة الاتجاه الأمر الذي يتطلب محاذاة المرسل مع المستقبل وجهاً لوجه.

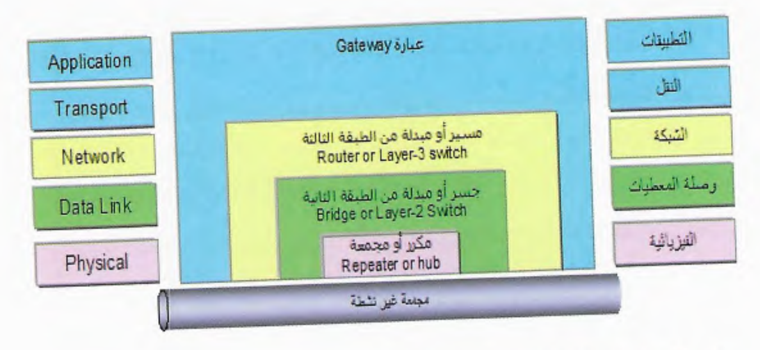
تفيد خاصية وحيد الاتجاه بتقليل التداخل الحاصل عند محاذاة عدة مرسلات ومستقبلات زوجاً زوجاً.

يوجد نوعين من الاتصالات الميكروية: أرضية للترددات الدنيا وعبر الأقمار الاصطناعية حيث ترسل المحطة إشارات إلى القمر الاصطناعي في النطاق *GHZ 6* بينما يعيد القمر الاصطناعي الإشارة إلى المحطة الأرضية بتردد ضمن النطاق *GHz 4*.

3 - 3 - الأجهزة الشبكية

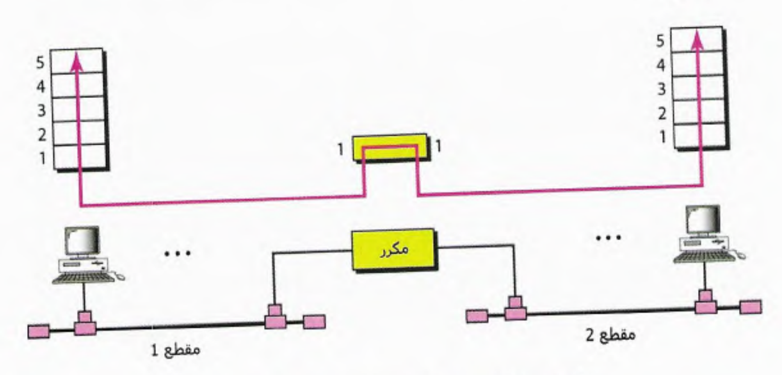
تقسم أجهزة ربط الشبكات إلى خمسة أصناف حسب المستوى الذي تعمل به. يبين الشكل *8* نوع كل جهاز ومستوى عمله.





3 - 3 - 1 - المكررات *Repeaters*

يعمل المكرر على المستوى الفيزيائي فقط. يقوم المكرر باستقبال الإشارات التي تحمل المعطيات وإعادة توليدها. تتعرض الإشارات أثناء انتقالها إلى التشوه أو التخامد. لذلك يفيد المكرر بزيادة الطول الفيزيائي لوسيط النقل أو الشبكة المحلية كما هو مبين في الشكل التالي.



الشكل -9 مثال عن مكرر يربط مقطعي شبكة محلية

تعريف: نسمي مقطع *Segment* شبكة محلية الجزء من الشبكة المحصور بين مكررين أو مكرر ومقاومة

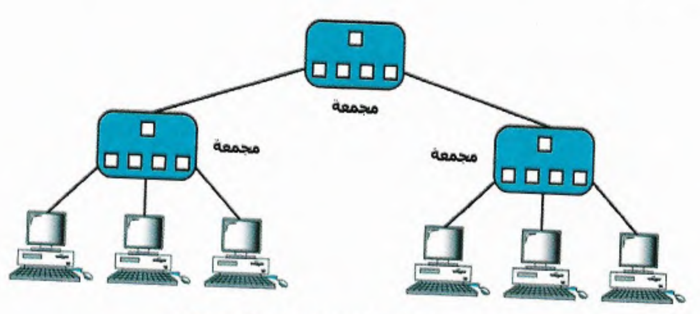
لاحظ أن المكرر لا يربط شبكتين محليتين وإنما يربط مقطعي شبكة محلية مع بعضهم البعض لتشكيل شبكة واحدة.

تتمثل وظيفة المكرر باستقبال الإشارات على بوابة ما وإعادة توليدها ومن ثم إرسالها على البوابة الأخرى بدون أي إمكانية تصفية.

يجب هنا التمييز بين المكرر والمضخم *Amplifier*؛ فالمضخم يقوم بتضخيم الإشارة والضجيج معاً بينما المكرر يعيد توليد الإشارة الأصلية كما كانت.

3 - 3 - 2 - المجمعة النشطة *Active Hub*

المجمعة النشطة هي مكرر متعدد البوابات. يجري استخدامها عادةً لتحقيق الوصل بين المحطات ضمن طوبولوجية نجمية. يمكننا أيضاً استخدام عدة مستويات من المجمعات كما هو موضح في الشكل التالي.

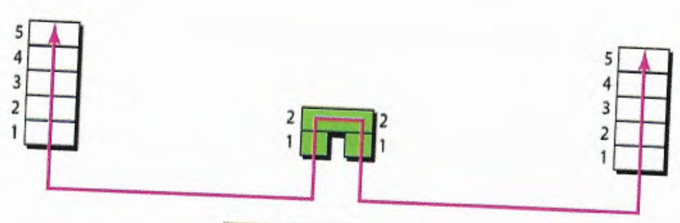


الشكل *-10* الاستخدام الهرمي للمجمعات

3 - 3 - 3 - الجسر *Bridge*

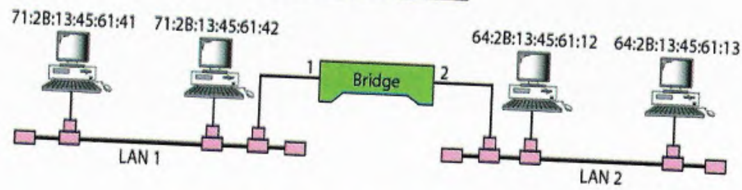
يعمل الجسر على المستويين الفيزيائي ووصلة المعطيات معاً. بالنسبة للمستوى الفيزيائي، فالجسر يقوم بإعادة توليد الإشارات المستقبلة؛ أما بالنسبة لمستوى وصلة المعطيات فإن الجسر يستطيع اختبار العناوين الفيزيائية *Physical MAC addresses* للمصدر وللوجهة الموجودين ضمن الإطار وأخذ القرار المتعلق بتوجيه الإطار أو إهماله؛ أي أن الجسر يستطيع **تصفية الأطر** *Frame Filtering* بناءً على العناوين الفيزيائية وفي حال وجود حاجة لتوجيهه فيجب هنا تحديد رقم البوابة التي سيجري التوجيه إليها. فإذاً، يمتلك الجسر جدول تقابل بين العناوين الفيزيائية وأرقام بوابات الجسر المرتبطة بها.

يبين الشكل التالي شبكتين محليتين مربوطتين إلى جسر. فإذا وصل إطار موجه إلى العنوان *71:2B:13:45:61:42* على البوابة رقم *1* فإن الجسر سيراجع جدول التقابل (أو جدول التوجيه) لتحديد بوابة الانطلاق. فحسب جدول التوجيه المتوفر لدى الجسر فإن الوصول إلى العنوان السابق يجري من خلال البوابة رقم *1* لذلك لا توجد حاجة لإعادة توجيه الإطار الذي يهمل. لكن إذا وصل إطار موجه إلى العنوان *71:2B:13:45:61:41* على البوابة رقم *2* حيث بوابة الانطلاق هي *1* فيقوم الجسر بإعادة توجيه الإطار على البوابة رقم *1*.



Address	Port
71:2B:13:45:61:41	1
71:2B:13:45:61:42	1
64:2B:13:45:61:12	2
64:2B:13:45:61:13	2

Bridge Table



الشكل -11 ربط شبكتين محليتين بواسطة جسر

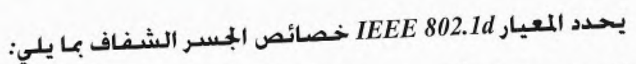
هذا يعني أنه في الحالة الأولى تم حصر حركة المرور في الشبكة الأولى بينما في الحالة الثاني فإن حركة المرور امتدت إلى الشبكة الثانية.

طبعاً يمكن للجسر أن يمتلك عدة بوابات في الوقت نفسه لكننا نرسم الجسر ببوابتين للتبسيط فقط.

يجب الانتباه هنا إلى أن الجسر لا يغير العناوين الفيزيائية الموجودة ضمن الأطر.

الجسور الشفافة *Transparent Bridges*

تعريف: نقول عن جسر أنه شفاف عندما تكون المحطات غير مدركة لوجوده؛ فلا نحتاج إلى إعادة إعداد المحطات في كل مرة نضيف أو نحذف جسر من الشبكة.



يحدد المعيار *IEEE 802.1d* خصائص الجسر الشفاف بما يلي:

- يجب توجيه الأطر من محطة إلى أخرى.
- تُبنى جداول التوجيه وتُحدث آلياً عن طريق التعلم الذاتي من حركة الأطر ضمن الشبكة.
- يجب تجنب حدوث حلقات ضمن الشبكة.

3 - 3 - 4 - مبدلات الطبقة الثانية *Layer-2 Switches*

يجب هنا التمييز بين نوعين من المبدلات: النوع الأول يعمل على المستوى الثاني (أي الطبقتين الفيزيائية ووصلة المعطيات) والنوع الثاني يعمل على المستوى الثالث (أي الطبقات الفيزيائية ووصلة المعطيات والشبكة).

تعريف: مبدلة الطبقة الثانية هي جسر متعدد البوابات مزود بإمكانيات معالجة متطورة.

يُستخدم الجسر عادةً لربط عدد محدد من الشبكات المحلية. لكن عندما يتوفر لدينا مبدلة بعدد كبير من البوابات فنستطيع ربط المحطة مباشرةً إلى المبدلة أو بشكل آخر تصبح كل شبكة محلية مكونة من المحطة وبوابة المبدلة.

تصفي المبدلة مثلها مثل الجسر الأطر حسب العناوين الفيزيائية كما يمكن تزويدها بذاكرة محلية *buffer* لتخزين الأطر لحين معالجتها.

3 - 3 - 5 - الموجهات *Routers*

يعمل الموجه على المستوى الثالث أي يقوم بتوجيه الطرود حسب العنوان المنطقي. يربط الموجه عادةً الشبكات المحلية مع الشبكات الواسعة ضمن الإنترنت ويحوي جدول توجيه يساعده على اتخاذ قرارات التوجيه اعتماداً على أفضل مسار (أو طريق) بين المصدر

والوجهة. تكون جداول التوجيه عادةً آلية التأقلم حيث يجري تحديثها باستخدام بروتوكولات التوجيه *Routing Protocols*.



الشكل *12-* استخدام الموجهات في الشبكات

3 - 3 - 6 - مبدلات الطبقة الثالثة *Layer-3 Switches*

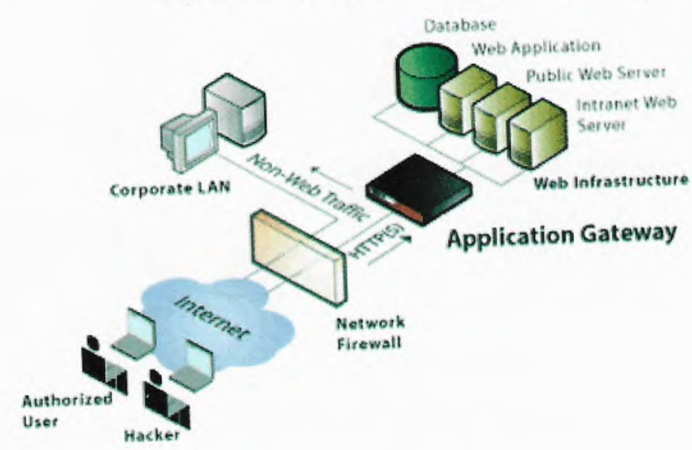
تمتاز المبدلة التي تعمل على المستوى الثالث عن الموجه بكونها أكثر سرعة وأكثر تطوراً. فالبنية الداخلية للمبدلة *Switching Fabric* تجعل عمليات البحث ضمن الجداول والتوجيه أكثر سرعة.

3 - 3 - 7 - العبّارة *Gateway*

تعمل العبارة ضمن جميع المستويات الخمسة بالنسبة للإنترنت والسبعة بالنسبة للنموذج *OSI*. تستقبل العبارة رسالة من تطبيق ما وتقرأه وتفسره ما يعني أننا يمكن استخدام العبارة لربط نظامين يستخدمان نموذجين مختلفين. لربط، على سبيل المثال، شبكة نظام يستخدم نموذج *OSI* المعياري ونظام آخر يستخدم نموذج الإنترنت.

تستطيع العبارة النفاذ إلى جميع الترويسات التي تولدها كل الطبقات وتوليد ترويسات مختلفة لذلك يطلق عليها في بعض الأحيان اسم محول بروتوكولات *Protocol Converter*.

يمكن أخيراً للعبارة أن تؤمن الحماية والأمن عن طريق تصفية الرسائل على مستوى التطبيقات *Application-level filtering*.



الشكل -13 مثال عن عبارة تطبيقات

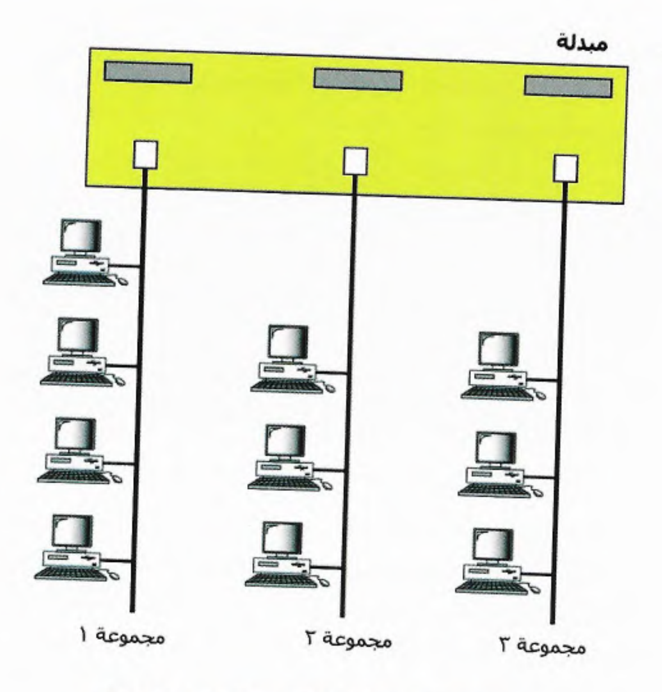
3 - 4 - الشبكات المحلية الافتراضية *VLANS*

تعتبر محطة ما جزءاً من شبكة محلية إذا كانت فيزيائياً مربوطة إلى تلك الشبكة. لذلك فعلاقة الانتماء هي علاقة جغرافية.

تعريف: الشبكة المحلية الافتراضية *Virtual LAN (VLAN)* هي شبكة محلية معرفة عن طريق البرمجيات وليس الطوبولوجية الفيزيائية.

لنأخذ المثال المبين في الشكل التالي:



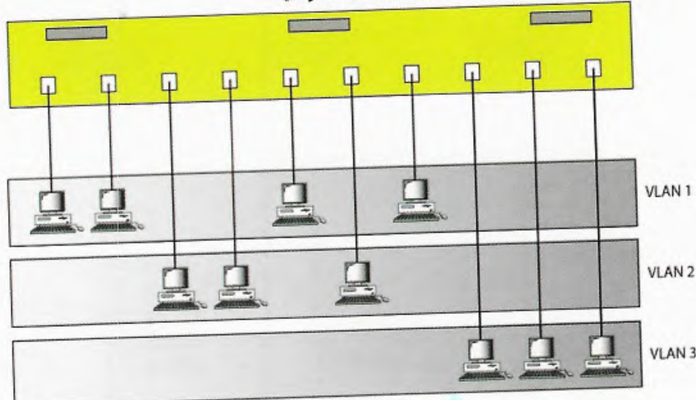


الشكل -14 ربط 3 شبكات محلية عن طريق مبدلة

لاحظ أننا مددنا الشبكة المحلية بطريقة تسمح بتعريف ثلاث مجموعات عمل بحيث يعمل أعضاء كل مجموعة سويةً أو يكون لديهم مشاريع مشتركة. لكن ماذا يحصل لو أراد مدير النظام نقل مهندسين من المجموعة الأولى إلى المجموعة الثالثة لتسريع العمل مثلاً؟ يجب على التقنيين إعادة تمديد الكابلات من جديد ومن ثم إعادة التمديد في كل مرة يطرأ تعديل على عضوية كل مجموعة.

يظهر الشكل التالي نفس الشبكات المحلية مقسمة إلى شبكات افتراضية.

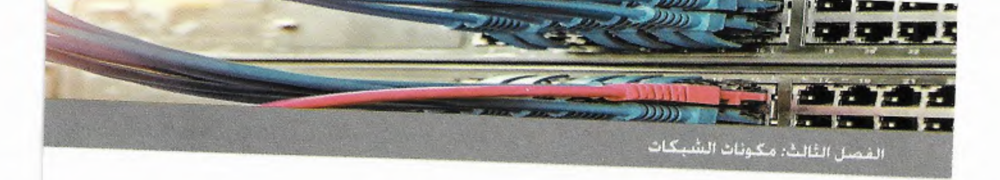




الشكل -15 تزويد المبدلة ببرمجيات لتحقيق الشبكات الافتراضية

تكمن الفكرة الأساسية خلف الشبكات الافتراضية بتقسيم الشبكة منطقياً وليس فيزيائيا. يدعى كل قسم منطقي بشبكة افتراضية وكل شبكة افتراضية تمثل مجموعة عمل أو قسم مختلف ضمن المؤسسة. لا نحتاج في هذه الحالة إلى تغيير التمديدات الفيزيائية للتأقلم مع انتقال الأشخاص من مجموعة عمل إلى أخرى وذلك لأننا نحدد عضوية الأشخاص إلى المجموعات برمجياً. لذلك نستطيع نقل كل محطة منطقياً من شبكة افتراضية إلى شبكة افتراضية أخرى.

عندما يرسل عضو من شبكة افتراضية تعميم*broadcast* فإن أعضاء هذه الشبكة الافتراضية فقط هم الذين يستقبلون الرسالة. أي أن الشبكات الافتراضية تسمح بتحسين أداء الشبكة عن طريق حصر رسائل التعميم على مستوى الشبكة الافتراضية وليس الشبكة ككل.

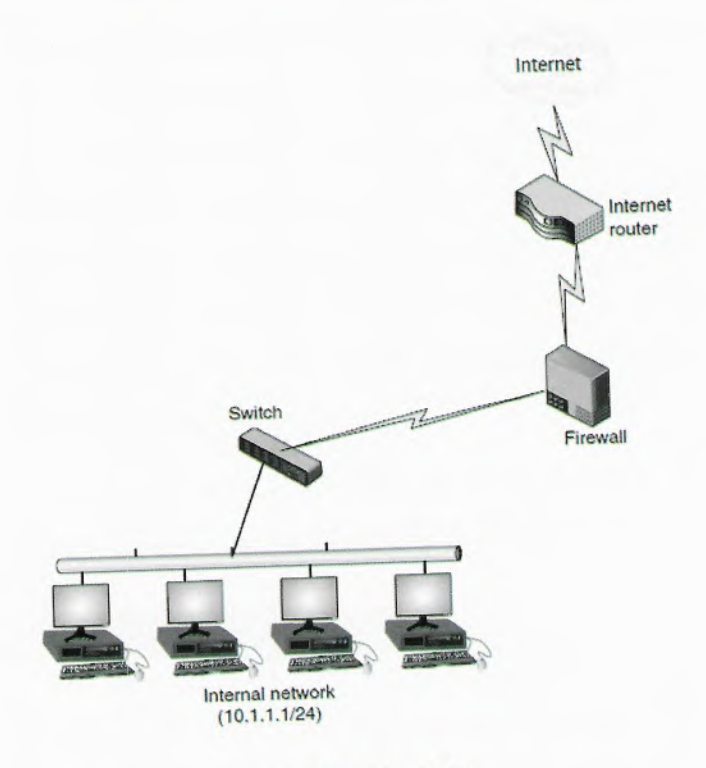


يتم تبادل الطرود ضمن شبكات *VLANS* بطريقتين. في حال كون المرسل والمستقبل موصولين إلى المبدلة نفسها فإن المبدلة تؤمن عملية تبادل الطرود بين الأجهزة التي تنتمي لشبكة افتراضية واحدة. أما في حال كون المرسل والمستقبل غير موصولين إلى المبدلة نفسها، فيجري استخدام بروتوكول وسم خاص الذي يمكن أن يكون مملوك من قبل شركة ما أو معياري مثل البروتوكول *IEEE 802.1Q*. يضيف هذا الأخير سمة (حقل بطول *16* بت) إلى الإطار تحمل رقم الشبكة الافتراضية للمرسل.

لا يمكن الاتصال بين جهازين ينتميان إلى شبكتين مختلفتين دون المرور ضمن موجه يحقق التوجيه بين الشبكات الافتراضية.

3 - 5 - جدران النار *Firewalls*

يوجد نوعين من جدران النار: **الشخصي والشبكي**. جدار النار الشخصي هو عبارة عن برنامج يعمل على الحاسوب المحلي لتصفية حركات المرور الشبكية من وإلى الحاسوب. أما جدار النار الشبكي فهو يصفي الطرود قبل دخولها إلى الشبكة. يقع جدار نار الشبكة عادةً خارج المحيط الأمني للشبكة ويعتبر خط الدفاع الأول كما هو موضح في الشكل التالي.



الشكل -16 توضع جدار النار

عندما يستقبل جدار النار طرداً ما فإنه إما يسمح له بالمرور أو يوقفه *Block* عن طريق تجاهله أو المطالبة (يسأل عن العملية المطلوب فعلها). يعمل جدار النار عادةً على أساس القواعد *Rule-based*. في هذه الحالة، فإنه يستخدم مجموعة من التعليمات الفردية للتحكم بالفعل. القاعدة هي مجموعة من المعلومات النصية التي تحدد عنوان الشبكة ورقم البوابة المسموح لهم / غير المسموح لهم بالدخول.



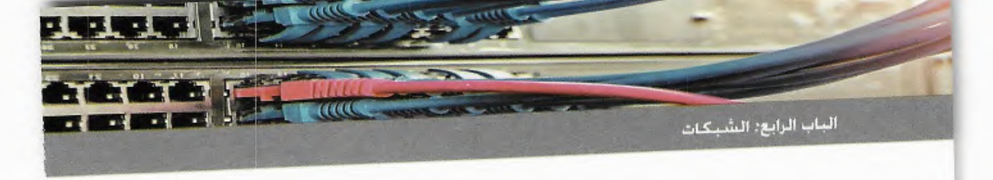
فمثلاً تسمح قاعدة ما لمستخدم من داخل الشبكة بإرسال طلب صفحة وب إلى مخدم وب خارجي. كما تسمح لمخدم الوب بإرسال الصفحة كجواب على الطلب السابق.

يلخص الجدول التالي بعض محتويات القواعد.

وصف القاعدة	شرح عنها	التصفية
Source address = any	عنوان *IP* للمخدم غير المعروف مسبقاً	لا يوجد تصفية لأننا لا نعرف عناوين *IP* للمخدمات بشكل مسبق
Destination address = internal IP address	عنوان *IP* للحاسوب الموجهة له الصفحة	تسمح هذه القاعدة للطرود الموجهة إلى عناوين محددة ضمن الشبكة المحلية بالمرور عبر جدار النار
Port = 80	هذا يعني أن البوابة رقم *80* مفتوحة	لا يوجد بوابات أخرى مفتوحة

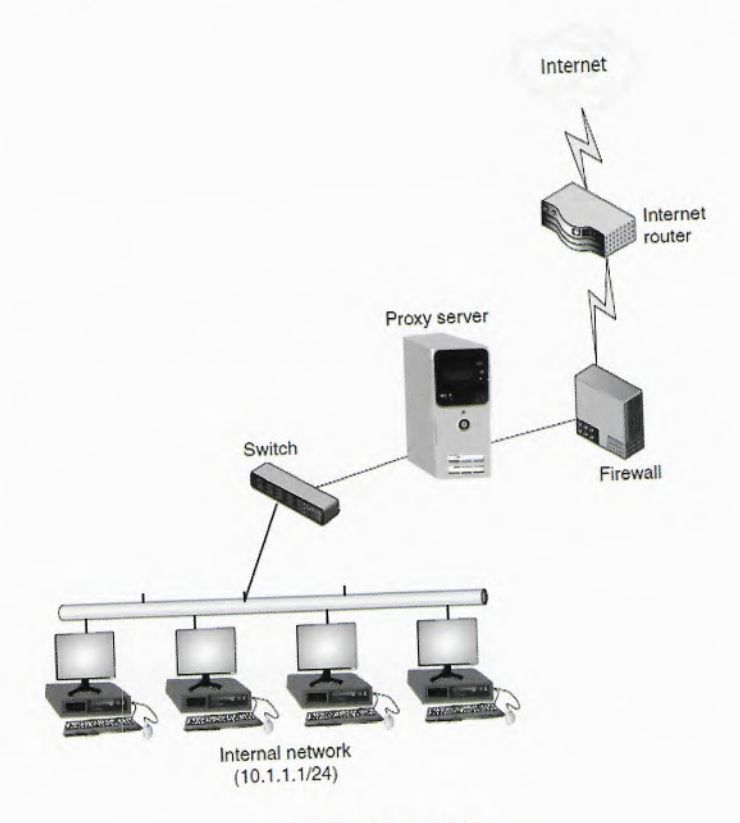
الجدول - *1* قاعدة مخصصة للاتصال مع مخدم وب

تحدد كل قاعدة لجدار النار ما يجب فعله مع كل طرد مستقبل. تخزن القواعد ضمن ملف أو عدة ملفات نصية يقرأها جدار النار عند الإقلاع. يعتبر هذا النوع من جدران النار ساكن وذلك لأن جدار النار لا يمكن أن يفعل شيئاً من خارج إعدادات القواعد الأمر الذي يجعله سهل الإعداد لكن غير مرن في التأقلم مع التغيرات.



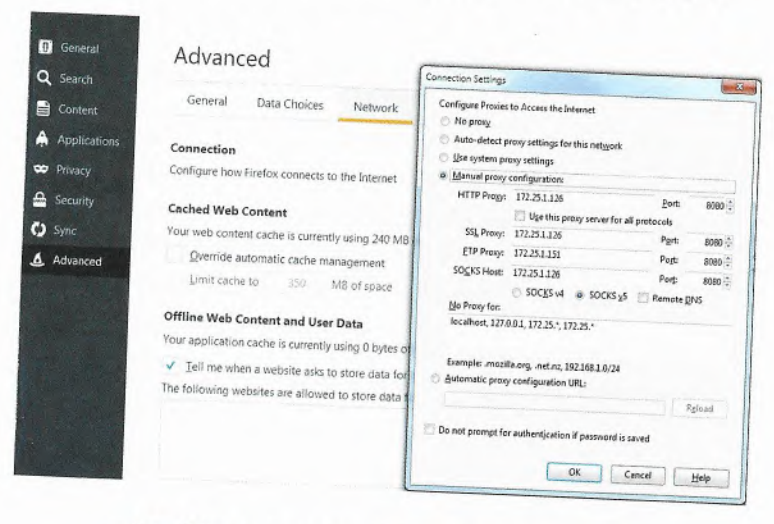
3 - 6 - المخدمات الوكيلة Proxies

المخدم الوكيل هو حاسوب أو برنامج تطبيقي يقوم باعتراض طلبات المستخدم القادمة من الشبكة الداخلية المحمية ومن ثم يعالج هذا الطلب بدلاً عن المستخدم. يوضح الشكل التالي آلية عمل المخدم الوكيل.

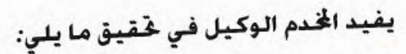


الشكل -17 المخدم الوكيل

عندما يطلب المستخدم الداخلي ملف ما أو صفحة وب من مخدم خارجي فإن الاتصال يكون مباشراً مع المخدم الخارجي في حال عدم وجود المخدم الوكيل. أما في حالة وجود المخدم الوكيل، فإن اتصال الزبون وإرسال الطلب يجري مع المخدم الوكيل الذي يختبر وجود جواب للطلب ضمن ذاكرة خابية *Cache* (أي يعمل المخدم الوكيل كمخدم تخبئة أيضاً). في حال وجود نسخة من الجواب ضمن الذاكرة الخابية فإن المخدم الوكيل يرسلها للمستخدم وإلا فإنه يتصل بمخدم الوب الخارجي باستخدام عنوان الإنترنت الخاص به ويطلب الصفحة. عندما يستقبل المخدم الوكيل الصفحة المطلوبة فإنه يعيد توجيهها إلى المستخدم بعد وضع نسخة منها ضمن الخابية. يجري إعداد الاتصال مع المخدم الوكيل من خلال متصفح الإنترنت كما هو موضح في الشكل التالي.



الشكل -*18* إعدادات المخدم الوكيل ضمن متصفح موزيلا فاير فوكس



يفيد المخدم الوكيل في تحقيق ما يلي:

- زيادة سرعة التحميل بسبب تخبئة المعطيات
- تخفيض التكاليف عن طريق تقليل سعة الوصلة المطلوبة
- تحسين الإدارة عن طريق توقيف مواقع الوب غير المرغوبة أو توقيف صفحات محددة منها
- تأمين حماية أمنية أعلى وذلك لأن المخدم الوكيل يخفي العنوان الحقيقي للزبون كما أنه يعمل كوسيط عند استقبال الأجوبة على طلبات المستخدمين أي أنه يستقبل أي برمجية خبيثة قبل أن تصيب المستخدم.

تجدر الإشارة إلى وجود مخدمات وكيلة شفافة لا يعلم المستخدم بوجودها ولا تتطلب أية إعدادات.

3 - 7 - كشف وتجنب التسلل *Intrusion Detection and Prevention*

يمكن الدفاع عن الشبكات بشكل فاعل *Active* أو بشكل سلبي *Passive*. الدفاع السلبي مثل استخدام جدار النار لتعطيل الهجمات على أساس القواعد الموضوعة أو الإعدادات الأمنية أو استخدام مصفي وب لتعطيل المواقع الخبيثة بينما الدفاع الفاعل مثل استخدام نظام كشف التسلل فهو قادر على كشف الهجوم أثناء وقوعه.

يمكن لأنظمة كشف التسلل استخدام منهجيات مختلفة لمراقبة الهجمات كما يمكن إعداد نظام كشف التسلل على مضيف محلي أو على الشبكة.

منهجيات المراقبة. تقتضي المراقبة اختبار حركات المرور الشبكية والنشاطات والسلوك لكشف حالات الشذوذ الأمنية. هناك أربع منهجيات مراقبة: المراقبة المعتمدة على الشذوذ والمراقبة المعتمدة على التوقيع والمراقبة المعتمدة على السلوك والمراقبة الإرشادية.

يجري تصميم المراقبة المعتمدة على الشذوذ *Anomaly-based monitoring* لكشف الشذوذ الإحصائي. يتم أولاً بناء خط أساس *Baseline* للنشاطات العادية خلال وقت محدد. عندما يلاحَظ انحراف كبير عن خط الأساس يجري تشغيل الإنذار أو التحذير المناسبين. من ميزات هذه المنهجية أنها تستطيع كشف الشذوذ بسرعة ودون الحاجة لمعرفة أسبابه. لكن هذه المنهجية تبقى عرضةً للإنذارات الخاطئة *False Positives* أي تلك الإنذارات التي تحدث دون وجود مبرر لها وذلك لأن النشاطات الطبيعية يمكن أن تتغير بسرعة أو مع مرور الزمن. إضافةً إلى الإنذارات الخاطئة، يمكن أن تسبب هذه المنهجية حمل معالجة كبير على النظام الذي يشغلها. أخيراً، بما أن هذه المنهجية تعتمد على بناء خط أساس، فهي تبقى عرضةً للهجمات طالما خط الأساس لم ينجز بعد.

أما المنهجية المعتمدة على التوقيع *Signature-based monitoring* (بصمة خاصة بكل هجوم) فتعتمد على التدقيق في حركات مرور الشبكة والنشاطات المختلفة والاتصالات والبحث عن نماذج معروفة وذلك بشكل قريب من آلية عمل مضادات الفيروسات. تتطلب هذه المنهجية النفاذ إلى قواعد معطيات للتواقيع محدثة بشكل متواصل إضافة إلى تعريف آلية المقارنة المطلوبة لاكتشاف التسلل. من سيئات هذه المنهجية كون قواعد معطيات التواقيع تزداد باستمرار وأن تعديل صغير في الهجوم يجعل توقيعه السابق غير مفيد.

تسعى المراقبة عن طريق السلوك *Behavior-based monitoring* إلى تجاوز حدود المنهجيتين السابقتين عن طريق العمل بشكل استباقي ومتأقلم عوضاً عن العمل بشكل تفاعلي. تأخذ هذه المنهجية الأعمال والإجرائيات العادية كقياس. تحلل هذه المنهجية الإجراءات والتطبيقات التي تعمل ضمن نظام ما بشكل متواصل وتحذر المستخدم عند اكتشاف أي فعل غير طبيعي ويقوم المستخدم بتعطيل أو بالسماح



بهذا الفعل. لا نحتاج هنا إلى تحديث قواعد معطيات التواقيع باستمرار أو بناء خط أساس إحصائي قبل البدء بالمراقبة. يفيد أيضاً هذا النوع من المراقبة في كشف الهجمات الحديثة.

تستند المراقبة الإرشادية *Heuristic monitoring* على تقنيات معتمدة على الخبرات *Experience-based techniques*. تحاول الإجابة على السؤال التالي: هل ستفعل هذه شيئاً ضاراً إذا سُمح لها بالتنفيذ؟ تستخدم هذه المنهجية خوارزمية لتحديد وجود تهديد.

أنواع أنظمة كشف التسلل

يوجد نوعان رئيسيان لأنظمة كشف التسلل وهما المضيف *HIDS* والشبكي *NIDS*. النظام المضيف *HIDS* هو تطبيق برمجي يعمل على مضيف محلي يستطيع كشف الهجوم أثناء حدوثه. يجري تنصيب المضيف عند كل حاسوب مطلوب حمايته سواء كان مكتبي أو مخدم. يعتمد المضيف على عملاء *Agents* مثبتة على النظام المطلوب حمايته. تعمل هذه العملاء بشكل وثيق مع نظام التشغيل لمراقبة واعتراض الطلبات في سبيل تجنب الهجمات. **تراقب *HIDS* عادةً الوظائف التالية:**

- استدعاءات النظام *System calls*. استدعاء النظام هو تعليمة تقاطع البرنامج قيد التنفيذ وتطلب خدمة من نظام التشغيل.
- النفاذ إلى نظام الملفات *File System access*. يتأكد أن جميع استدعاءات طلب فتح الملفات هي شرعية وليست نتيجة نشاطات خبيثة.
- إعدادات تسجيلات النظام *System Registry settings*. يراقب *HIDS* سجلات نظام *Windows* التي تحوي معلومات التكوينات المتعلقة بالبرامج والحاسوب ويتعرف على أي تعديل غير مسموح به.

- دخل / خرج المضيف *Host input / output*. يراقب *HIDS* كل اتصالات الدخل والخرج بحثاً عن أي نشاط خبيث. فمثلا، إذا كان نظام ما لا يستخدم الرسائل الفورية، وفي لحظة ما حاول النظام تنفيذ اتصال *IM* فإن *HIDS* سيكشف هذا الخطر على أنه نشاط شاذ.

نذكر من سيئات *HIDS* أنه غير قادر على مراقبة حركات المرور الشبكية غير الموجهة للنظام المراقب وأنه يخزن جميع سجلات الأثر *Log file* بشكل محلي كما أنه يسبب ضغطاً كبيراً على الموارد المحلية الأمر الذي يسبب بطء النظام.

على الجانب الآخر، يراقب نظام كشف التسلل الشبكي *NIDS* الهجمات على الشبكة. يجري هنا وضع حساسات *Sensors* على بعض التجهيزات الشبكية مثل الموجهات أو جدران النار لتجميع المعلومات عن حركات المرور الشبكية وتوليد التقارير عنها وإرسالها إلى جهاز مركزي.

نذكر من التقنيات التي يمكن أن يستخدمها *NIDS*:

- التحقق من مكدس البروتوكولات *Protocol stack verification* عن طريق كشف طرود *IP, TCP, UDP, ICMP* غير النظامية
- التحقق من بروتوكولات التطبيقات *Application protocol verification* عن طريق التحقق من بعض سلوكيات البروتوكولات غير الصحيحة
- خلق سجلات موسعة *extended logs* عن طريق تسجيل الأحداث غير العادية بغية تمريرها لمحلل التسجيلات.

ما أن يتم كشف الهجوم، يمكن لنظام *NDIS* التصرف بطرق مختلفة. يقوم النظام السلبي *passive NDIS* بتشغيل إنذار وتسجيل الحدث ضمن *Log*. يمكن أن يشمل الإنذار إرسال بريد الكتروني أو رسالة *SMS*



لمدير الشبكة أو قرع إنذار صوتي. أما النظام الفاعل *active NIDS* فيقوم إضافة إلى ما سبق بفعل ما. يمكن أن يتمثل الفعل في إعادة تشكيل جدار النار لتعطيل الطرود القادمة من عنوان إنترنت محدد أو تشغيل برنامج آخر لمعالجة الحادث أو إنهاء جلسة *TCP*.

يمكننا اعتبار أن نظام تجنب التسلل الشبكي *NIPS* مشابه لنظام كشف تسلل شبكي فاعل من حيث كشف الهجوم واتخاذ ما يلزم لتعطيله. لكن الفارق الرئيسي بينهما هو مكان التواجد. فنظام *NIDS* يملك حساسات تراقب حركات المرور الداخلة والخارجة من جدار النار وتبلغ جهاز مركزي للتحليل. أما نظام *NIPS* فهو يوجد في نسق جدار النار نفسه الأمر الذي يسمح له بالقيام بفعل سريع لتعطيل الهجوم.



3 - 8 - الأسئلة

1. يكون / تكون _____ شفافاً.

   a. الكابل المحوري

   b. الأزواج المجدولة

   c. <u>الألياف الضوئية</u>

   d. الكابلات التسلسلية

2. تعتبر _____ محصنة كلياً ضد التداخلات الكهرطيسية

   a. الكابلات المحورية

   b. الأزواج المجدولة

   c. <u>الألياف الضوئية</u>

   d. الوصلات اللاسلكية

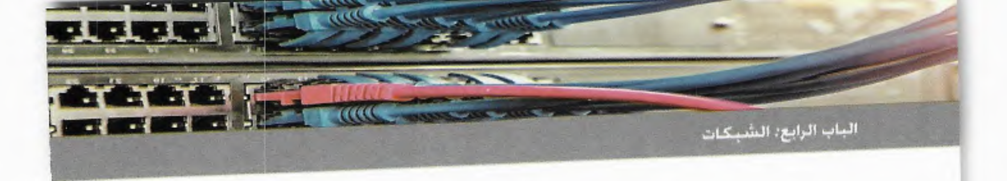
3. يستطيع _____ تصفية الأطر بناءً على العناوين الفيزيائية.

   a. المكرر

   b. <u>الجسر</u>

   c. الموجه

   d. العبارة



4. يعتبر ـــــ و ـــــ من أشهر أنواع كابلات الأزواج الجدولة.

a. STP و UDP

b. **<u>STP و UTP</u>**

c. FB و STP

d. TCP و UDP

5. يفيد ـــــ في زيادة الطول الفيزيائي لوسيط النقل فقط.

a. **<u>المكرر</u>**

b. الموجه

c. الجسر

d. العبارة

6. يعرف المعيار ـــــ الجسور الشفافة.

a. 802.3

b. 802.1

c. **<u>802.1d</u>**

d. 802.5



7. يعمل ـــــ على توجيه الطرود ضمن الإنترنت اعتماداً على عنوان *IP*.

a. المكرر

b. الجسر

c. <u>الموجه</u>

d. مبدلة الطبقة الثانية

8. تنقل كابلات الأزواج المجدولة الإشارات على شكل ـــــــــ.

a. أمواج كهرطيسية

b. <u>تيار كهربائي</u>

c. أشعة تحت حمراء

d. أشعة مايكروية

9. الموصل الأشهر لكابلات الأزواج المجدولة هو ـــــــــ.

a. *RJ11*

b. <u>RJ45</u>

c. *RG45*

d. *RG11*

10. تدعى الأمواج التي ترددها محصور بين ______ بالأمواج الميكروية.

a. ***<u>GHz and 300 GHz 1</u>***

b. *MHZ and 300 MHz 1*

c. *THZ and 300 THz 3*

d. *GHz and 400 GHZ 300*

11. يعمل المكرر *repeater* على مستوى ______.

a. طبقة التطبيقات

b. طبقة النقل

c. طبقة الشبكة

d. **<u>الطبقة الفيزيائية</u>**

12. تعمل العبّارة *Gateway* على مستوى ______.

a. طبقة التطبيقات

b. طبقة النقل

c. طبقة الشبكة

d. **<u>الطبقات السبعة</u>**



13. لا تعتبر ______ من وسائط النقل الموجهة.

a. كابلات الأزواج المجدولة

b. كابلات الألياف الضوئية

c. الأمواج الميكروية

d. الكابلات المحورية

14. يرمز لكابلات الأزواج المجدولة غير المحمية بـ ______.

a. *UDP*

b. *UTP*

c. *STP*

d. *FTP*

15. يستخدم الموصل *BNC* مع ______.

a. الأزواج المجدولة

b. الكابلات المحورية

c. الألياف الضوئية

d. الكابلات التسلسلية

16. لا تعتبر ______ من طرق انتشار الأمواج الراديوية.

a. **الأفقي**

b. الأرضي

c. الفضائي

d. وفق خط نظر

17. يعمل هذا الجهاز على مستوى طبقة الشبكة.

a. المكرر

b. الجسر

c. **الموجه**

d. العبارة

18. لا تتمتع الشبكات المحلية الافتراضية *VLANs* بهذه الخاصية.

a. تعرف برمجياً

b. تقلل التعميم

c. تحتاج لبروتوكول وسم الطرود

d. **تقسم الشبكة فيزيائياً**



19. يعمل جدار النار عادةً اعتماداً على ______.

a. الأحداث

b. **<u>القواعد</u>**

c. السماحيات

d. الامتيازات

20. أي من التالي ليس لديه طبقة ربط معطيات *(Data-Link)*

a. *Router*

b. *Gateway*

c. *switch*

d. *bridge*

e. ***<u>repeater</u>***





الفصل الرابع

شبكة إيثرنت المحلية

Ethernet Local Area networks (LANs)

تعريف- الشبكة المحلية *Local Area Network (LAN)*: هي شبكة حواسيب يجري تصميمها لتخديم مسافة جغرافية محدودة كبناء أو حرم.

مع أنه يمكن استخدام الشبكة المحلية داخل مؤسسة ما بشكل معزول لتحقيق المشاركة على الموارد إلا أنه يجري وصل غالبية الشبكات المحلية الحالية إلى الشبكات الواسعة أو إلى الإنترنت.

لقد عرفت الأسواق أنواعاً عدة من الشبكات المحلية مثل الإيثرنت *Ethernet*، والحلقة ذات العلام *Token Ring*، والممر ذو العلام *Token Bus*، وشبكات الألياف الضوئية،*FDDI* وشبكات النقل غير المتزامن *ATM LAN*. بقي بعض هذه الشبكات حياً لفترة من الزمن إلا أن شبكة إيثرنت بقيت هي التقانة المهيمنة حتى وقتنا هذا.

سنخصص هذه الفصل لشبكة إيثرنت التي شهدت تطوراً ملحوظاً مثلاً بأربعة أجيال خلال العقد الأخير من الزمن اقتضته الحاجة إلى مواكبة حركة ومتطلبات السوق، غير أن المفاهيم الرئيسة لشبكة إيثرنت بقيت نفسها.



4 - 1 - معايير *IEEE*

بدأ، خلال عام *1985*، معهد *IEEE* مشروعاً أطلق عليه اسم *Project 802* مهمته وضع المعايير التي تسمح بربط تجهيزات من موردين مختلفين. لم يكن الهدف من هذا المشروع استبدال النموذج المرجعي *OSI* وإنما كُلِف بتوصيف وظائف الطبقة الفيزيائية وطبقة وصلة المعطيات للبروتوكولات المستخدمة ضمن الشبكات المحلية.

يبين الشكل *1* العلاقة بين النموذج المرجعي *OSI* ومعايير *IEEE 802*.

LLC : Logical Link Control
MAC : Media Access Control

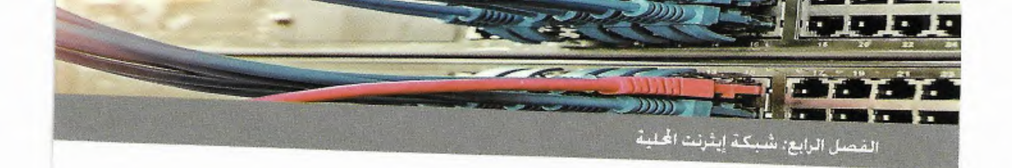
الطبقات العليا			
التحكم بالوصلة المنطقية LLC			
إيثرنت MAC	الحلقة ذات العلام MAC	الممر ذو العلام MAC	....
الطبقة الفيزيائية	الطبقة الفيزيائية	الطبقة الفيزيائية	....
وسيط النقل			

معايير IEEE

النموذج OSI
الطبقات العليا
طبقة وصلة المعطيات
الطبقة الفيزيائية
وسيط النقل

الشكل *1*- معايير *IEEE* للشبكات المحلية

لاحظ أن معهد *IEEE* قسم طبقة وصلة المعطيات إلى طبقتين جزئيتين: التحكم بالوصلة المنطقية *Logical Link Control (LLC)* والتحكم بالنفاذ إلى الوسيط *Media Access Control (MAC)*. قام معهد *IEEE* أيضاً بتعريف طبقات فيزيائية مختلفة لكل نوع من أنواع الشبكات المحلية.



4 - 1 - 1 - التحكم بالوصلة المنطقية LLC

تزود هذه الطبقة الجزئية بروتوكولاً وحيداً للتحكم بطبقة وصلة المعطيات من أجل جميع الشبكات المحلية، بعكس التحكم بالنفاذ إلى الوسيط المتعلق بالشبكة المحلية. يساعد توحيد LLC في تسهيل عملية ربط شبكات محلية مختلفة فيما بينها وذلك لأنه يخفي جميع الاختلافات الموروثة من طبقة MAC.

تتمثل العمليات الأساسية التي تقوم بها طبقة LLC في التأطير والتحكم بالأخطاء وبالتدفق. نحتاج إلى هذا النوع من التحكم ضمن الشبكات المحلية المعزولة لكن عندما نستخدم طبقات عليا مثل TCP/ IP، فلا تعود هناك حاجة لهذا النوع من التحكم.

4 - 2 - التحكم بالنفاذ إلى الوسيط MAC

تعرف طبقة MAC الجزئية طرق التحكم بالوسيط لكل نوع من أنواع الشبكات المحلية. فشبكة إيثرنت تستخدم بروتوكول التنصت على الحامل مع النفاذ المتعدد وكشف التصادم Carrier Sense Multiple Access with Collision Detection (CSMA / CD) بينما تستخدم الحلقة ذات العلام طريقة أخرى تتمثل بمرور علام على الشبكة والمحطة التي تمتلك العلام تستطيع الإرسال.

تقوم أيضاً هذه الطبقة الجزئية بمعالجة جزء من وظائف التأطير.

4 - 3 - طبقة التحكم بالنفاذ IEEE 802.3 MAC

تعتمد طبقة التحكم بالنفاذ على بروتوكول CSMA / CD. يعمل هذا البروتوكول وفق الخطوات التالية:

1. تتنصت المحطة التي تريد الإرسال إلى الوسيط (أو الحامل). فإذا كان حراً (لا توجد إشارات عليه) فإنها تقوم بالإرسال وإلا فإنها تنتقل إلى الخطوة الثانية.

2. إذا كان الوسيط مشغولاً فإن المحطة تستمر بالتنصت حتى يصبح حراً عندها تقوم المحطة بالإرسال مباشرةً.

3. عندما يتم، أثناء الإرسال، اكتشاف حالة تصادم مع الإشارات التي تولدها محطة أو محطات أخرى، فتقوم المحطة بإرسال إشارة التشويش *Jamming Signal* تضمن فيها أن جميع المحطات تصبح قادرة على اكتشاف التصادم ومن ثم توقف عملية الإرسال.

4. تنتظر المحطة، بعد إرسال إشارة التشويش، زمناً عشوائياً، يطلق عليه اسم *Backoff Time*، ثم تحاول الإرسال مجدداً بالعودة إلى الخطوة *1*.

يبين الشكل *2* كيفية حصول التصادم على ممر موصول إليه أربع محطات.

الشكل -*2* آلية عمل بروتوكول *CSMS/CD*

1. في اللحظة $t0$، تبدأ المحطة A بإرسال إطار إلى المحطة D.

2. في اللحظة $t1$، تكون المحطتان B وC جاهزتين للإرسال. تتنصت B على الوسيط فترى وجود إشارة عليه فتؤجل العملية. لكن C، حتى اللحظة، لا تتحسس وجود أي إشارة على الوسيط (لأن الإشارة التي أرسلتها A لم تصل بعد إلى C) فتقوم بالإرسال.

3. عندما تصل الإشارات المرسلة من قبل A إلى C في اللحظة $t2$، فإن الأخيرة تتحسس وجود تصادم وتوقف عملية الإرسال.

4. يصل أثر التصادم إلى A في اللحظة $t3$ حيث توقف A عملية الإرسال أيضاً.

لاحظ أنه باستخدام $CSMA / CD$، يكون الزمن اللازم لاكتشاف التصادم هو الوقت الضائع بدون فعالية.

ما هو هذا الزمن؟ لنفترض أننا مازلنا نستخدم طوبولوجية ممرية، وأن المحطتين تقعان في طرفي الممر (أي A وD بالنسبة للشكل 1). فإذا افترضنا أن A بدأت الإرسال، وقبل أن تصل إشاراتها إلى D بدأت تلك الأخيرة بالإرسال أيضاً. يقع التصادم وتتحسس D مباشرةً لكنه يجب على هذه الإشارات المشوهة الناتجة عن التصادم أن تنتشر على طول الكابل حتى تصل إلى A وتتحسس التصادم. ينتج مما سبق أن الزمن اللازم لاكتشاف التصادم هو أصغر من ضعفي زمن انتشار الإشارة على طول الكابل من نهاية لنهاية.

قاعدة أساسية ضمن بروتوكول $CSMA / CD$: يجب أن يكون الإطار ذو طول كبير بحيث يقع التصادم قبل أن تنته المحطة من إرساله (أو بمعنىً آخر، عندما تنته محطة ما من إرسال إطار ولا تتحسس وجود تصادم فإن المحطة تعتبر أن الإطار وصل بسلام).



4 - 3 - 1 - صيغة إطار *MAC*

يبين الشكل *3* صيغة إطار بروتوكول *Ethernet MAC*.

				46 to 1500 octets			
7 octets	1	6	6	2	≥ 0	≥ 0	4
Preamble	SFD	DA	SA	Length	LLC Data	Pad	FCS

SFD : Start of Frame Delimiter
DA : Destination Address
SA : Source Address
FCS : Frame Check Sequence

الشكل -*3* صيغة إطار معيار *IEEE 802.3*

يتألف الإطار من الحقول التالية:

المهد *Preamble*: هو عبارة عن سبعة بايتات مؤلفة من واحدات وأصفار متعاقبة تستخدم لتحذير المحطة المستقبلة إلى الإطار القادم وتمكنها من مزامنة مؤقتها الزمني مع مؤقت المحطة المرسلة.

محرف تحديد بداية الإطار *Start Frame Delimiter (SFD)*: وهو عبارة عن بايت واحد *(10101011)* يدل على بداية الإطار. عندما لا يكون المستقبل متزامناً مع المرسل فإنه من الممكن أن يضيع بعض البتات فتصبح الطريقة الوحيدة لاكتشاف بداية الإطار هي بوجود واحدين متعاقبين.

عنوان الوجهة *DA (Destination Address)*: وهو عبارة عن *6* بايتات يحوي على العنوان الفيزيائي للمحطة الوجهة أو المحطات الموجه إليها الإطار.

عنوان المصدر *Source Address (SA)*: وهو يتألف أيضاً من *6* بايتات ويحوي على العنوان الفيزيائي للمرسل.

الطول أو النوع *Type or Length*: استخدمت إيثرنت الأصلية هذا الحقل للدلالة على نوع بروتوكول الطبقة العليا الذي يستخدم الإطار. أما معايير *IEEE* فتستخدم هذا الحقل للدلالة على الطول أو عدد البايتات الموجودة ضمن حقل المعطيات.

المعطيات *LLC Data*: يحوي هذا الحقل المعطيات المغلفة من بروتوكول طبقة *LLC* العليا. يجب أن تكون المعطيات ضمن المجال *(46 bytes – 1500 bytes)*.

الحشوة *Pad*: مجموعة من البايتات التي تضاف إلى المعطيات حتى يصبح طول المعطيات *44* بايتاً على الأقل دون حقل *Length*.

تسلس فحص الإطار *FCS*: والذي يستخدم اختبار التكاملية *CRC*، وهي معلومات خاصة باكتشاف الأخطاء على جميع الحقول باستثناء *Preamble, SFD, and FCS* حيث يتم استخدام *CRC-32*.

طول الإطار

يعود سبب تحديد الطول الأصغري لإطار إيثرنت إلى حاجة بروتوكول *CSMA/CD* الذي يتطلب إطاراً ذو طول *64* بايت مع الترويسة واللاحقة. وبما أن الترويسة واللاحقة، دون حقل *Preamble and SFD*، يشغلان *18* بايتاً فيصبح الطول الأصغري للمعطيات هو *46* بايت. فإذا كانت المعطيات القادمة من الطبقة الأعلى أقل من *46* بايت فيجب إضافة حشوة *Padding* لملء الفراغ.

أضف إلى ذلك أن المعايير تحدد الطول الأعظمي للإطار، بدون *Preamble and SFD*، بـ *1518 bytes* فإذا حذفنا منها *18* بايت فيبقى لدينا *1500 bytes* تشكل الطول الأعظمي للمعطيات القادمة من الطبقة العليا.

العنونة *Addressing*

تحوي كل محطة مربوطة إلى شبكة إيثرنت بطاقة شبكة *Network Interface Card (NIC)*. تمتلك كل بطاقة، كما رأينا سابقاً، شبكة عنوان فيزيائي مكون من *6* بايتات تكتب كالمثال التالي:

06 : 01 : 02 : 01 : 2C : 4B

العناوين وحيدة الوجهة ومتعددة الوجهات والمعممة

يكون عنوان المصدر دائماً وحيد الوجهة *Unicast* وذلك لأن الإطار يأتي دائماً من محطة واحدة فقط. أما بالنسبة لعنوان الوجهة فيمكن أن يكون وحيد الوجهة أو متعدد الوجهات *Multicast* أو معمم *Broadcast*.

يمكن التمييز بين هذه الأنواع المختلفة عن طريق البت الأقل دلالة *Least Significant bit* من البايت الأول (إلى اليسار) لعنوان الوجهة: فإذا كانت قيمة البت *0* فالعنوان وحيد الوجهة وإذا كانت القيمة *1* فالعنوان متعدد الوجهات كما هو موضح في الشكل التالي:

الشكل -4 العناوين وحيدة الوجهة ومتعددة الوجهات

يعرف العنوان وحيد الوجهة مستقبل وحيد أي أن العلاقة بين المرسل والمستقبل هي علاقة واحد إلى واحد. أما العنوان متعدد الوجهات فهو يعرف مجموعة من المستقبلين وتصبح العلاقة بين المرسل والمستقبل هي علاقة واحد إلى مجموعة.

بالنسبة لعنوان التعميم (أو العنوان المعمم) فهو حالة خاصة من العنوان متعدد الوجهات حيث يكون المستقبلون هم جميع المحطات الموجودة على الشبكة وتكون جميع بتات العنوان مساوية *1* أي *All ones*.

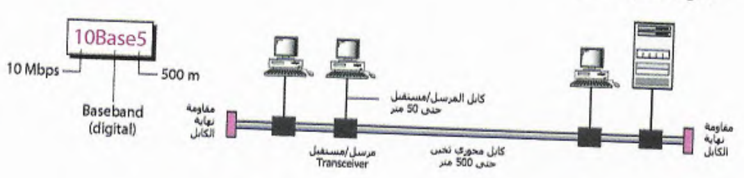
يجب الانتباه هنا إلى أن إرسال البايتات يجري من اليسار إلى اليمين (بايتاً بايتاً) وبالنسبة لكل بت داخل البايت فيجري من اليمين إلى اليسار (البت الأقل دلالة أولاً). لذلك فإن البايت *4A* يرسل أولاً في الحالة الأولى كما يجري إرسال بت تحديد نوع العنوان ضمن هذا البايت أولاً.

4 - 4 - شبكة إيثرنت IEEE 10 Mbps

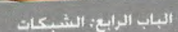
عرفت لجنة *IEEE 802.3* مجموعة من المواصفات الفيزيائية للطوبولوجيات التي يمكن استخدامها:

4 - 4 - 1 - إيثرنت الثخينة 10Base5

تعود التسمية إلى ثخانة الكابل المستخدم لتحقيقها. لقد كانت شبكة إيثرنت الثخينة أول شبكة إيثرنت تستخدم طوبولوجية ممرية مع مرسل / مستقبل *Transceiver* خارجي يتم ربطه عن طريق فرعة *Tap* إلى كابل محوري ثخين. يبين الشكل التالي مخططاً لهذا النوع من الشبكات.

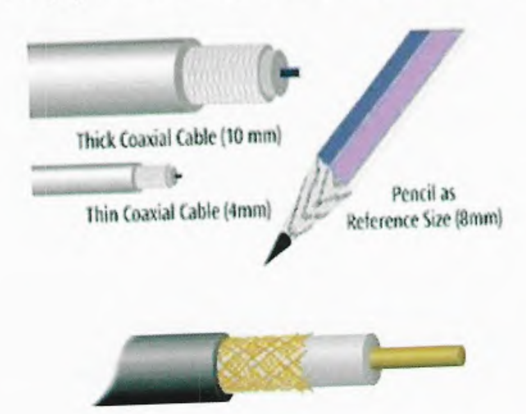


الشكل -5 تحقيق إيثرنت الثخينة 10Base5



تتمثل مسؤولية المرسل / مستقبل في النقل والاستقبال واكتشاف التصادم. يجري وصل المرسل / مستقبل على محطة عن طريق كابل خاص بالمرسل / مستقبل يؤمن مسارين منفصلين للإرسال وللاستقبال، ما يعني أن التصادمات يمكن أن تقع على الكابل المحوري فقط.

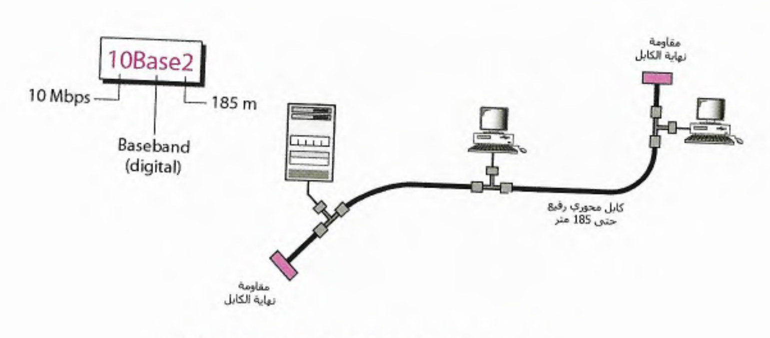
يجب أن لا يزيد طول الكابل المحوري عن *500* متر وإلا فإن الإشارات ستتشوه نتيجةً للتخامد. غير أنه من الممكن استخدام *4* مكررات (أو *5* مقاطع شبكة) على الأكثر ما يوسع قطر الشبكة إلى *2500* متر.



الشكل *-6* صورة للكابل المحوري الثخين

4 - 4 - 2 - إيثرنت الرفيعة *10Base2*

تستخدم إيثرنت الرفيعة أيضاً الطوبولوجية المرية لكن الكابلات المحورية تكون أقل ثخانةً وأكثر مرونةً لذلك يمكننا ثنيها بحيث تستطيع المرور بالقرب من المحطات. يصبح، في هذه الحالة، المرسل / مستقبل جزءاً من بطاقة الشبكة الموضوعة ضمن المحطة. يبين الشكل التالي مخططاً لهذا المعيار.



الشكل -7 تحقيق شبكة إيثرنت الرفيعة *10Base2*

يعتبر تحقيق هذا النوع من الشبكات أقل كلفةً من النوع السابق وتمديده أسهل. لكن طول الكابل ينخفض إلى *185* متر بسبب مستوى التخامد الكبير الذي يعاني منه.



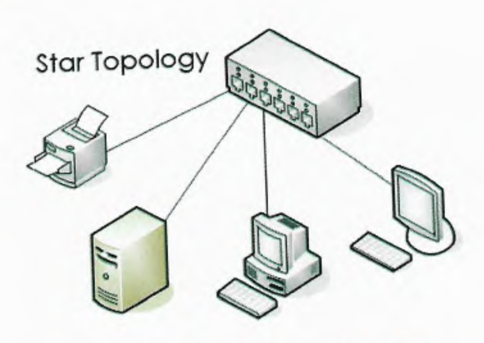


الشكل -8 الكابلات المحورية الرفيعة ونهاياتها وتوصيلاتها

4 - 4 - 3 - إيثرنت الأزواج المجدولة *10Base-T*

يستخدم هذا النوع من شبكات إيثرنت طوبولوجية نجمية حيث يجري ربط جميع المحطات إلى مجمعة مركزية *Hub* باستخدام زوجين مجدولين كما هو موضح في الشكل التالي.

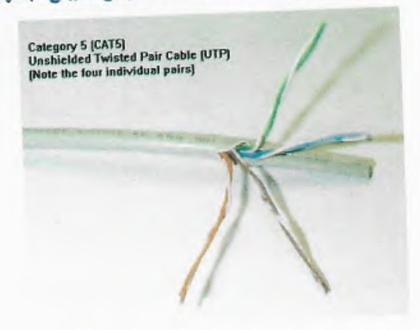




الشكل -9 تحقيق إيثرنت الأزواج المجدولة *10Base-T*



الشكل -10 موصل *RJ-46* المستخدم مع الأزواج المجدولة



الشكل -11 الأزواج المجدولة غير المحمية



لاحظ أن الزوجين المجدولين يشكلان مسارين (الأول للإرسال والثاني للاستقبال) بين المحطة والمجمعة؛ لذلك فإن التصادمات تحدث ضمن المجمعة.

يحدد طول الأزواج المجدولة بين أي محطة وبين المجمعة بـ *100* متر لتقليل أثر التخامد قدر الإمكان.

4 - 4 - 4 - إيثرنت الضوئية *10Base-F*

تضم ثلاث مواصفات: طوبولوجية نجمية خاملة *Passive-star topology* لربط محطات ومكررات لمسافة تصل إلى *1 km* للمقطع الواحد؛ وصلات نقطة لنقطة تستخدم لربط محطات أو مكررات لمسافة تصل إلى *2 km*؛ وصلة نقطة لنقطة تستخدم لوصل المكررات لمسافة تصل إلى *2 km*.



الشكل *-12* الألياف الضوئية



ويلخص الجدول التالي مواصفات كل نوع من الأنواع السابقة:

	10BASE5	10BASE2	10BASE-T	10BASE-FP
وسيط النقل	كابل محوري 50 أوم	كابل محوري 50 أوم	أزواج مجدولة غير محمية *Unshielded Twisted Pair (UTP)*	ألياف ضوئية *850-nm*
الطوبولوجية	ممرية	ممرية	نجمية	نجمية
الطول الأعظمي للمقطع (متر)	500	185	100	500
عدد المحطات في المقطع	100	30	-------	33
قطر الكابل (مم)	10	5	0.4 to 0.6	62.5/125 µm

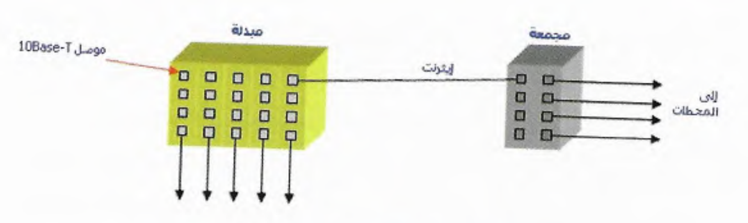
الجدول 1 - خصائص الطبقة الفيزيائية لمعيار IEEE 802.3

4 - 4 - 5 - إيثرنت المبدلة Switched Ethernet

إن الإضافة المستمرة للمحطات إلى شبكة إيثرنت تؤدي إلى زيادة التصادمات وبالتالي اختناق الشبكة. تسعى تقانة تبديل إيثرنت على معالجة مسائل حركات المرور المتزايدة على الشبكة المحلية.

يبين الشكل التالي مبدلة تحوي بطاقة حاملة *Backplane* عالية السرعة وفتحات لاستيعاب عدداً من البطاقات *Line Cards*. يحوي كل منها عدد يتراوح بين *1* و *8* موصلات. يملك كل موصل غالباً وصلة من نوع *10Base-T* إلى محطة عمل.





الشكل -13 مثال بسيط عن إيثرنت المبدلة

عندما تريد محطة ما إرسال إطار إيثرنت، فإنها توجه هذا الإطار إلى المبدلة. تقوم البطاقة المركبة *Plug-in card* على المبدلة التي تستقبل الإطار باختبار إمكانية الوصول إلى عنوان الوجهة من خلال البطاقة نفسها. في حال تحقق الشرط فإنها تنسخ الإطار على الموصل المربوط إلى للمحطة الوجهة. في حال عدم تحقق الشرط، فتقوم البطاقة بإرسال الإطار عن طريق البطاقة الحاملة إلى البطاقة الموصول إليها المحطة الوجهة. تعمل البطاقة الحاملة بسرعات عالية تصل إلى عدة جيغا بت بالثانية.

لكن ماذا يحدث لو أرسلت محطتين، مربوطتين إلى نفس البطاقة المركبة، في نفس الوقت؟ يتعلق هذا بطريقة تصميم البطاقة المركبة. ففي حال جرى وصل جميع بوابات البطاقة معاً لتشكيل شبكة محلية على البطاقة فإن البطاقة ستعمل كمجمعة إيثرنت عادية مع احتمال وقوع تصادمات وحلها باستخدام بروتوكول *CSMA / CD*. أي تستطيع محطة واحدة ضمن كل بطاقة الإرسال في لحظة ما مع إمكانية أن تعمل جميع البطاقات على التوازي. أي أن كل بطاقة تشكل مجال تصادم منفصل عن بقية المحطات.

أما بالنسبة للنوع الثاني من البطاقات المركبة فيجري تخزين كل الأطر القادمة على كل بوابة ضمن ذاكرة *RAM* مدمجة مع للبطاقة. هذا يسمح لبوابات الدخل أو الخرج بالاستقبال أو الإرسال في نفس

الوقت ما يتيح العمل بالاتجاهين *Full-duplex*. بعد أن يتم استقبال الإطار بشكل كامل، تقوم البطاقة باختبار وجهتها: إذا كانت موجهة إلى عنوان محلي (أي على نفس البطاقة) فيجري إرسالها مباشرةً إلى بوابة الوجهة. أما إذا كانت موجودة على بوابة بعيدة (أي على بطاقة أخرى) فيجري إرسالها عن طريق البطاقة الحاملة *Backplane*. يصبح لدينا، باستخدام هذا التصميم، مجال تصادم لكل بوابة أو بشكل آخر لا يصبح لدينا أي تصادم.

لا يوجد ما يمنع وصل بوابة من بوابات المبدلة إلى مجمعة خارجية طالما أن كل بوابة من بوابات المبدلة تتوقع استقبال أطر إيثرنت.

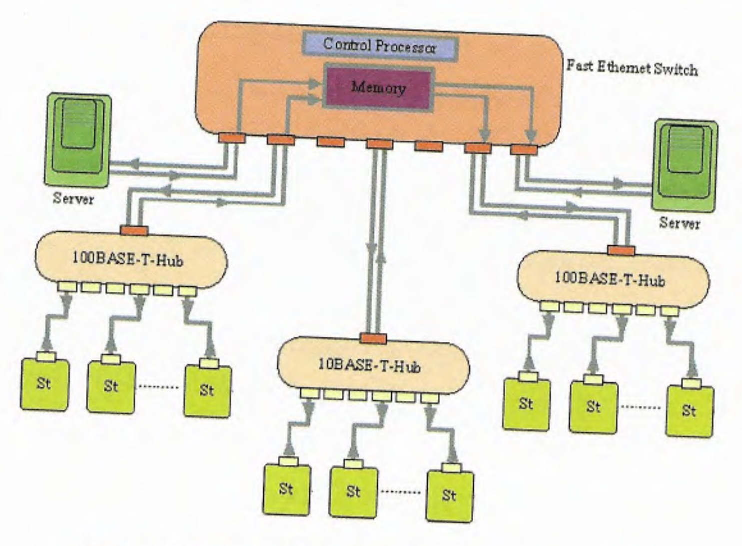
إيثرنت مزدوجة الاتجاه *Full-duplex Ethernet*

لعل أحد مساوئ شبكات *10Base5* و *10Base2* هو كونهم وحيدي الاتجاه؛ يمكن لمحطة ما أن ترسل أو تستقبل في لحظة ما ولكن ليس كليهما. لذلك، سعى التطور الأول إلى جعل إيثرنت المبدلة مزدوجة الاتجاه الأمر الذي يرفع استطاعة كل مجال تصادم إلى *20 Mbps* بدلاً عن *10 Mbps*. يجري تحقيق ذلك باستخدام وصلتين بين المحطة والمبدلة: الأولى للإرسال والثانية للاستقبال.

لاحظ أنه عند استخدام إيثرنت المبدلة مزدوجة الاتجاه، لا تعود هناك حاجة لاستخدام بروتوكول *CSMA / CD*، وذلك لعدم إمكانية حدوث أي تصادم، فكل وصلة بين المبدلة والمحطة هي مجال تصادم قائم بذاته وثنائي الاتجاه.

يبين الشكل التالي مثالاً عن ربط مجمعات وحيدة الاتجاه مع مبدلة مزدوجة الاتجاه.





الشكل *14*- ربط مجمعات إيثرنت وحيدة الاتجاه إلى مبدلة مزدوجة الاتجاه

4 - 4 - 6 - إيثرنت السريعة *Fast Ethernet*

يوصف المعيار *IEEE 802.3u* الذي وضع عام *1995* خصائص شبكة إيثرنت السريعة. تتميز شبكة إيثرنت السريعة بكونها متوافقة مع إيثرنت التقليدية لكن بمعدل نقل معطيات أكبر بعشر مرات. **يمكننا تلخيص أهداف شبكة إيثرنت السريعة على النحو التالي:**

- زيادة معدل نقل المعطيات إلى *Mbps 100*
- جعلها متوافقة مع إيثرنت المعيارية
- المحافظة على نفس العناوين الفيزيائية المكونة من *bits 48*
- المحافظة على صيغة الإطار نفسها
- المحافظة على نفس الأطوال الصغرى والعظمى للإطار.



1 - طبقة *MAC* الجزئية

جرى أثناء تصميم طبقة *MAC* الجزئية لإيثرنت السريعة استبعاد الطبولوجيات الممرية والاحتفاظ فقط بالطوبولوجية النجمية. يبقى لدينا خياران بالنسبة للطوبولوجية النجمية: نصف المزدوج والمزدوج. يجري في الحل نصف المزدوج وصل المحطات إلى مجمعة مركزية بينما في الحل المزدوج فيجري وصل المحطات إلى مبدلة مزودة بدارئة *Buffer* عند كل بوابة.

يبقى بروتوكول التحكم بالنفاذ *CSMA / CD* نفسه في الحل نصف المزدوج بينما يمكن توقيفه في الحل المزدوج.

التفاوض الذاتي *Auto-negotiation*

يسمح التفاوض الذاتي لجهازين بالتفاوض على نمط أو معدل نقل المعطيات. لقد جرى تصميمه بشكل خاص لتحقيق ما يلي:

- السماح بوصل أجهزة غير متوافقة مع بعضها البعض. أي يمكن وصل جهاز يعمل بسرعة *10 Mbps* مع جهاز آخر يعمل بسرعة *100 Mbps* (عن طريق تخفيض سرعته إلى *10 Mbps*).
- السماح لجهاز واحد بامتلاك معدلات نقل معطيات مختلفة.
- السماح لمحطة ما باختبار مقدرات المجمعة.

2 - الطبقة الفيزيائية

الطوبولوجية

يمكن استخدام شبكة إيثرنت السريعة لربط محطتين أو أكثر: يجري ربط محطتين فقط عن طريق كابل مباشر ومعكوس بين المحطتين. في حال وجود أكثر من محطتين فإننا نحتاج إلى مجمعة أو مبدلة لتحقيق الربط النجمي.



التحقيق

يمكن تحقيق إيثرنت السريعة على المستوى الفيزيائي باستخدام سلكين أو أربعة أسلاك. يجري استخدام السلكين مع الأزواج المجدولة غير المحمية *Unshielded Twisted Pairs (UTP) Category 5* المستخدمة في *100Base-TX* أو الكابلات الضوئية *(100Base-FX)*. بينما يجري استخدام أربعة أسلاك مع كابلات *UTP Category 3* فقط *(100BaseT4)*.

يبين الشكل التالي الكابلات التي تستخدمها إيثرنت السريعة.

الاسم	الكابلات	الطول الأعظمي للمقطع	الحسنات
100Base-T4	الأزواج المجدولة	*100* متر	تستخدم كابلات *UTP Cat. 3*
100Base-TX	الأزواج المجدولة	*100* متر	مزدوجة الاتجاه مع *UTP Cat.5*
100Base-FX	الألياف الضوئية	*2000* متر	مزدوجة الاتجاه للمسافات الطويلة

الجدول 2 - الكابلات التي تستخدمها إيثرنت السريعة

4 - 4 - 7 - الجيغابت إيثرنت *Gigabit Ethernet*

تعمل شبكة *Gigabit Ethernet*، التي عرفت باسم *IEEE 802.3z*، بمعدل نقل معطيات يصل إلى *1000 Mbps* أو *1 Gbps*. جرى طبعاً المحافظة على جميع الأهداف التصميمية التي ذكرناها ضمن إيثرنت السريعة مع إضافة التوافق مع إيثرنت السريعة ودعم التفاوض الذاتي.

طبقة *MAC* الجزئية

يعرف المعيار *IEEE 802.3z* الذي تم تصديقه عام *1998* شبكة الجيغا بت إيثرنت. تدعم الجيغابت إيثرنت نمطي الإرسال أحادي وثنائي الاتجاه. مع أن غالبية شبكات إيثرنت العاملة بسرعة *1 Gbps* تتبع نمط الإرسال ثنائي الاتجاه إلا أننا سنناقش نمط الإرسال أحادي الاتجاه المعمول به من أجل المحافظة على التوافق مع الأجيال السابقة.

نمط الإرسال المزدوج

نستخدم في هذا النمط مبدلة إيثرنت مربوطة إلى جميع المحطات أو بقية المبدلات. تمتلك كل مبدلة دارئة لكل بوابة دخل تخزن فيها الأطر لحين إرسالها. لا يوجد تصادمات ضمن هذا النوع من الإرسال ولا يتم استخدام بروتوكول *CSMA / CD*. بالنسبة للقيمة العظمى لطول الكابل فيحدده تخامد الإشارة ضمن الكابل وليس بروتوكول التحكم بالنفاذ.

نمط الإرسال نصف المزدوج

يجري هنا استخدام مجمعة مركزية تقلد عمل الكابل حيث يمكن حدوث التصادمات، لذلك يجب استخدام بروتوكول *CSMA / CD*. يتعلق الطول الأعظمي للشبكة، في هذه الحالة، بالطول الأصغري للإطار. لذلك فقد جرى تعريف ثلاثة أنماط عمل: تقليدي وتوسيع الحامل ورشق الأطر.

نمط العمل التقليدي: نحافظ في هذا النمط على طول الإطار الأصغري كما هو عليه الحال في إيثرنت التقليدية (أي *512 Bits*). لكن في هذا النمط يصبح قطر الشبكة مساوٍ لـ *25* متر لأننا نرسل بسرعة أكبر بمائة مرة من إيثرنت التقليدية فعدد البتات الموجود على الشبكة يصبح أكبر بمائة مرة فيجب تخفيض قطر الشبكة مائة مرة



(من *2500* متر حتى *25* متر). يمكن قبول هذا النمط إذا كانت جميع التجهيزات موجودة ضمن غرفة واحدة لكنه يبقى غير مناسب في الحالة العامة.

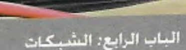
نمط توسيع الحامل *Carrier Extension*: يجري هنا زيادة طول الإطار الأصغري ليصبح *512 bytes* بدلاً عن *64 Bytes*. أي أننا ضاعفنا القيمة ثمان مرات. يجبر هذا النمط كل محطة على إضافة حشوة *Padding* إلى أي إطار أصغر من *512 Bytes*. بهذه الطريقة يمكننا مضاعفة الطول الأصغري للشبكة ثمان مرات ليصبح *200 m* مع الاحتفاظ بمسافة *100 m* بين المجمعة والمحطة (علل لماذا ؟)

رشق الأطر *Frame Bursting*: جرى طرح هذا النمط لمعالجة عدم فعالية النمط السابق خاصةً في حالة الأطر الصغيرة. فبدلاً عن توسيع الطول الأصغري للإطار يمكننا إرسال عدة أطر معاً (أي ضمن إطار واحد) بعد إضافة حشوات بين الأطر حتى لا تصبح القناة خاملة. أو بشكل آخر، خداع المحطات الأخرى بجعلها تعتقد أننا قيد إرسال إطار كبير وليس عدة أطر.

تبقى الطوبولوجية الفيزيائية كما كانت عليه في إيثرنت السريعة. أما بالنسبة لتحقيق الجيغابت إيثرنت فيمكننا تصنيفه أيضاً عن طريق استخدام سلكين أو أربعة أسلاك.

التقانات التي تحقق الجيغابت إيثرنت باستخدام سلكين فقط:

- *1000Base-SX (Short-wave) Fiber-optic cable*
- *1000Base-LX (Long-wave) Fiber-optic cable*
- *1000Base-CX (Shielded Twisted Pair) STP cable*



التقانة التي تستخدم أربعة أسلاك هي تقانة الأزواج المجدولة غير المحمية Cat. 5e UTP أو أعلى 1000Base-T.

يبين الشكل التالي مقارنة بين الأنواع السابقة للكابلات.

الاسم	الكابلات	الطول الأعظمي للمقطع	الحسنات
1000Base-SX	الألياف الضوئية	550 متر (50 ميكرون)	ألياف متعددة الأنماط (50 and 62.5 μm)
1000Base-LX	الألياف الضوئية	5000 متر (أحادي النمط)	ألياف أحادية النمط أو متعددة الأنماط
1000Base-CX	زوجي STP	25 متر	الأزواج المجدولة المحمية
1000Base-T	أربعة أزواج UTP	100 متر	الأزواج المجدولة غير المحمية

الشكل -15 الكابلات التي تستخدمها الجيغا بت إيثرنت

لاحظ أنه يمكننا استخدام كابلات ضوئية وفق ثلاثة أقطار مختلفة: 10, 50, 62.5 μm. الأول (أي 10) لأحادية النمط والباقي لمتعددة الأنماط.



4 - 4 - 8 - العشرة جيغابت إيثرنت *Ten-Gigabit Ethernet*

تعرف العشرة جيغابت إيثرنت باسم *IEEE 802.3ae*. لقد جرى تصميمها بشكل خاص لتحقيق ما يلي:

- رفع معدل نقل المعطيات إلى *10 Gbps*.
- جعلها متوافقة مع إيثرنت التقليدية والسريعة والجيغابت إيثرنت.
- المحافظة على نفس طول العناوين الفيزيائية وهي *48 bits*.
- المحافظة على نفس صيغة الإطار.
- المحافظة على القيمة الدنيا والعليا لأطوال الإطارات
- السماح بتحقيق ترابط الشبكات المحلية الموجودة باستخدام شبكة إقليمية *MAN* أو واسعة *WAN*
- جعل إيثرنت متوافقة مع تقانات مثل مرحل الأطر *Frame Relay*، والنقل غير المتزامن *ATM (Asynchronous Transfer mode)*.

تعمل شبكة العشرة جيغابت إيثرنت بنمط النقل المزدوج فقط مع استبعاد بروتوكول *CSMA/CD*.



4 - 5 - الأسئلة

1. تعرف الشبكات المحلية وفق المصطلح ______.

a. *MAN*

b. *WAN*

c. <u>*LAN*</u>

d. *SAN*

e. *PAN*

2. يستخدم هذا الحقل ضمن إطار إيثرنت لاكتشاف الأخطاء.

a. *Preamble*

b. *DA*

c. *SA*

d. <u>*FCS*</u>

e. *SFD*

3. بروتوكول النفاذ إلى الوسيط التي تستخدمه شبكة إيثرنت هو ______.

a. *Token Passing*

b. *CSMA/CA*

c. <u>*CSMA/CD*</u>

d. *Polling*

e. *TCP*



4. هذا الحقل غير موجود ضمن إطار إيثرنت.

a. *Destination address*

b. *Source address*

c. *Length*

d. <u>*Port number*</u>

e. *CRC*

5. إذا استقبلت طبقة التحكم بالنفاذ ضمن إيثرنت طرداً ذو طول 20 بايت، فإنها ستضيف حشوة بطول ــــــ.

a. 10 بايت

b. 20 بايت

c. <u>24 بايت</u>

d. 64 بايت

e. 52. بايت

6. يصل طول مقطع شبكة إيثرنت وفق هذا المعيار إلى 500 متراً.

a. *10Base-T*

b. <u>*10Base-5*</u>

c. *10Base-2*

d. *10Base-F*

e. *5Base-T*



7. تستخدم شبكة ______ أزواجاً مجدولة.

a. **<u>10Base-T</u>**

b. *10Base-5*

c. *10Base-2*

d. *10Base-F*

e. *5Base-F*

8. تستخدم شبكة ______ أربعة أسلاك.

a. *100Base-TX*

b. **<u>100Base-T4</u>**

c. *100Base-FX*

d. *100Base-T*

e. ولا إجابة من السابق.

9. قسم معهد *IEEE* الشبكات المحلية إلى.

a. *MAC and IP*

b. **<u>MAC and LLC</u>**

c. *HDLC and MAC*

d. *Ethernet and IP*

e. *HDLC and IP*



10. يتألف العنوان الفيزيائي من.

a. *4* بايتات

b. *8* بايتات

c. *16* بايت

d. <u>***6* بايتات**</u>

e. *12* بايت

11. لا يعتبر ____ من معايير إيثرنت العاملة بسرعة *10 Mbps*.

a. *10base-2*

b. <u>***10base-C***</u>

c. *10base-5*

d. *10base-F*

e. *10base-T*

12. تعمل إيثرنت السريعة بمعدل نقل معطيات يساوي _____.

a. *Mbps 10*

b. <u>***Mbps 100***</u>

c. *Gbps 1*

d. *Gbps 10*

e. *Gbps 100*



13. لا يستخدم هذا المعيار مع الألياف الضوئية.

a. *1000base-SX*

b. *1000base-LX*

c. ***<u>1000base-CX</u>***

d. كل الإجابات السابقة

e. ولا إجابة من السابق.

14. بين تسلسل إرسال العنوان:

EE : 08 : 2E : 1B : 20 : 47 :

على وسيط النقل.

<u>الحل</u>

11011000 *00000100* *¬ 11100100*

01110111 *00010000* *01110100*

15. ما هو الطول الأعظمي لجزء *100Base-FX*؟

a. ***<u>m 2000</u>***

b. *m 100*

c. *m 185*

d. *m 5000*

e. *m 200*



16. عدد البايتات في عناوين الإيثرنت لشبكات *CSMA/CD* هو:

a. 16

b. 5

c. <u>6</u>

d. 48

e. 4

17. نمط الاتصال الذي يدعم معطيات في كلا الاتجاهين هو:

a. *Simplex*

b. <u>Duplex</u>

c. *Half Duplex*

d. *Multiplex*

e. ولا إجابة من السابق.

18. تستخدم شبكة *(FDDI)* الطبولوجيا الفيزيائية التالية:

a. *Bus*

b. <u>Ring</u>

c. *Star*

d. *Tree*

e. *Mesh*



19. ترمز (*FDDI*) إلى:

a. <u>*Fiber Distributed Data Interface*</u>

b. *Fiber Data Distributed Interface*

c. *Fiber Dual Distributed Interface*

d. *Fiber Distributed Data Internetwork*

f. ولا إجابة من السابق.

20. هدف خانات التمهيد (*preamble bits*) في إطار الإيثرنت هو:

a. تهيئة *AR.*

b. تحديد عدد خانات.

c. التمهيد

d. <u>التزامن</u>

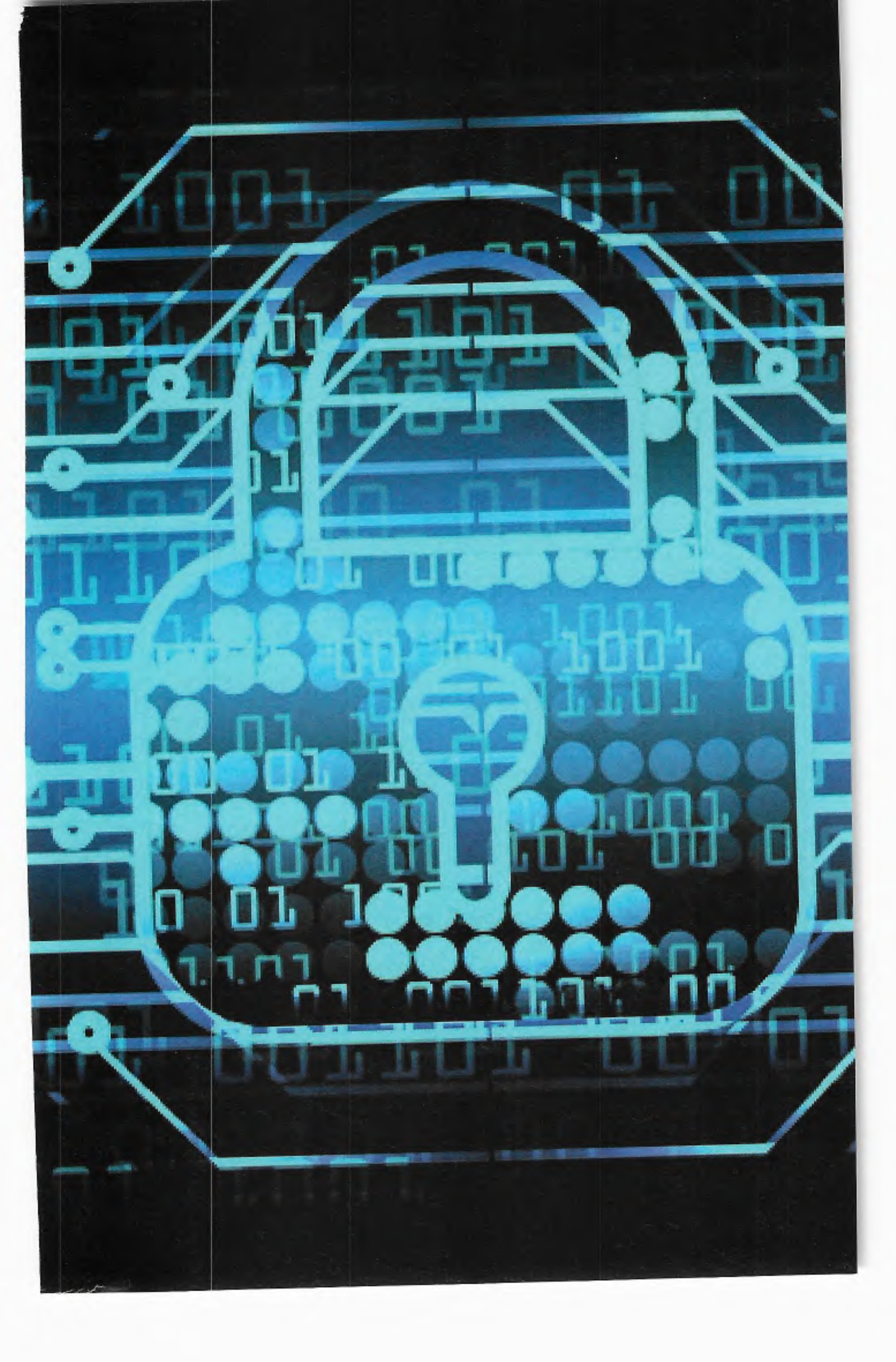
e. فحص الأخطاء

f. عنوان الهدف.

الباب الخامس

أمن النظم المعلوماتية

(Information System Security)



أمن النظم المعلوماتية

Information System Security

الهدف من هذا المقرر إظهار أهمية الأمن للمعلومات والنظم المعلوماتية ونظم الاتصالات. إضافة إلى إعطاء نظرة واسعة لحقل أمن المعلومات والأمور المرتبطة به وتعريف مصطلحات هذا الحقل.

سنتناول في هذا الباب المواضيع التالية:

الفصل الأول: مدخل إلى أمن المعلومات والنظم المعلوماتية

- مكونات النظام المعلوماتي التي يشملها الأمن
- تعريف الأمن *Security*
- الأهداف المفتاحية لأمن المعلومات والنظم المعلوماتية
- مجالات أمن النظم المعلوماتية
- إدارة أمن النظم المعلوماتية

الفصل الثاني: أدوات التعمية

- تعاريف
- أنواع التعمية
- المشفرات التقليدية – التشفير بالمفتاح العشوائي الوحيد
- المشفرات المتناظرة التسلسلية
- المشفرات المتناظرة الكتلية

- إدارة مفاتيح التشفير
- خوارزمية التشفير بالمفتاح العمومي
- مقارنة بين التعمية المتناظرة والتعمية بالمفتاح العمومي
- خوارزميات التحقق من الرسالة
- التوقيع الرقمي

الفصل الثالث: التحقق من المستخدم

- وسائل التحقق من المستخدم
- التحقق المستند إلى كلمة المرور
- تقنية كلمة المرور المهشرة
- التحقق باستخدام البطاقات الذكية
- التحقق باستخدام المزايا البايومترية
- التحقق عند الدخول عن بعد

الفصل الرابع: مهددات النظم المعلوماتية

- مهددات النظم المعلوماتية
- أنواع الهجوم
- البرمجيات الخبيثة

الفصل الخامس: أمن الشبكات

- نظام كشف الاختراق
- الجدار الناري
- نظام منع الاختراق

الفصل السادس: بروتوكولات أمن الانترنت

- أمن البريد الالكتروني باستخدام بروتوكول *S / MIME*
- طبقة المقابس الآمنة *SSL*
- بروتوكول الإنترنت الآمن *IPSec*

الفصل السابع: الأمن الفيزيائي

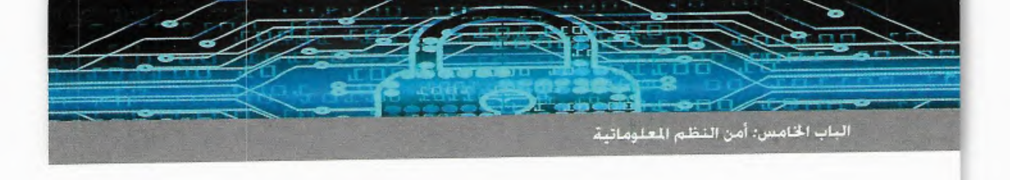
- مهددات الأمن الفيزيائي
- التعافي من خروقات الامن الفيزيائي
- التكامل بين الأمن الفيزيائي والأمن المنطقي

الفصل الثامن: إدارة الأمن – المعايير

- تعريف إدارة الأمن
- وظائف إدارة الأمن
- معايير إدارة أمن النظم المعلوماتية
- السياسة الأمنية

الفصل التاسع: الجرائم المعلوماتية

- تعريف الجريمة المعلوماتية
- الجرائم المعلوماتية التقليدية
- الجرائم المعلوماتية المستحدثة
- قانون" تنظيم التّواصل على الشّبكة ومكافحة الجريمة المعلوماتيّة"



الفصل الأول

مدخل إلى أمن المعلومات والنظم المعلوماتية

Introduction to Information System Security

1 - 1 - مكونات النظام المعلوماتي التي يشملها الأمن

يتكون النظام المعلوماتي من الممتلكات *assets* التالية:

- البيانات والمعلومات *data / information*
- العتاديات *hardware*، الحاسوب وتوابعه من وحدات تخزين ومحيطيات
- البرمجيات *software* بكافة أنواعها سواء أكانت أساسية أم تطبيقية
- الشبكات وخدمات الاتصالات *networks and communication facilities*

يضاف إلى الممتلكات:

- الإجرائيات التي تمكن من استخدام المعلومات كمصادر في المؤسسة
- الأشخاص: مدير النظام والمستخدمون

1 - 2 - تعريف الأمن *Security*

نقصد بالأمن في هذا السياق حالة كون النظام المعلوماتي خال من أي مخاطر. أي حماية كل مكونات النظام المعلوماتي المعرفة أعلاه من أي خطر. وهناك ثلاث مصطلحات تستخدم في هذا السياق هي:

- **المهدد *threat*:** يُعرف بأنه كائن أو شخص يشكل خطراً مستمراً على الممتلكات.
- **الضعف *vulnerability*:** ضعف محدد في النظام المعلوماتي الخاضع للرقابة أو الحماية. مثل كون النظام المعلوماتي قابل للخرق أو يتضمن نظام حماية غير فعال او مُحدث *updated*.
- **الهجوم *attack*:** الفعل الذي يستغل الضعف في النظام الخاضع للحماية أو الرقابة.

المهددات موجودة باستمرار، بينما الهجمات هي أفعال تؤدي إلى أضرار بالنظام المعلوماتي.

1 - 3 - الأهداف الرئيسية لأمن المعلومات والنظم المعلوماتية

تهدف الإجراءات الأمنية إلى تحقيق خمسة أهداف مفتاحية إثنان منها حديثان، وهي:

- **السرية *confidentiality*:** وهي خاصية منع كشف المعلومات لأشخاص غير مخولين بالاطلاع عليها من خلال تعميتها.
- **سلامة المعلومات *integrity*:** الحفاظ على المعلومات (خلال النقل أو التخزين) دون تعديل من أشخاص غير مخولين بذلك.
- **الإتاحية *availability*:** توفر المعلومات والتطبيقات عند طلبها لتمكن النظام المعلوماتي من أداء دوره (ضمان الوصول للمعلومات).

ومع ظهور شبكات الإنترنت واستخدام البريد الإلكتروني أو المعاملات الإلكترونية ظهر هدفان جديدان هما:

- **المساءلة *accountability*:** نقصد من هذا الهدف الأمني توفير الإجراءات التي تمكننا من تحديد المسؤوليات عند القيام بعمل باستخدام النظام المعلوماتي. ومن ضمنها مايسمى "عدم

الإنكار" *non-repudiation* بحيث نوفر الأدوات التي تمنع مثلاً المرسل أو المستقبل لمعاملة ما من إنكار ذلك.

- **الاستيقان** ***authenticity:*** خاصية أن يكون حقيقياً مع إمكانية التحقق والثقة، أي الثقة في صحة نقل الرسالة أو الرسالة نفسها أومن أنشأ هذه الرسالة.

1 - 4 - مجالات أمن النظم المعلوماتية

هناك مجالات / طبقات من الأمن في النظام المعلوماتي هي:

- أمن البيانات / المعلومات
- أمن الشبكات والاتصالات
- الأمن الفيزيائي (أمن البنية التحتية والتجهيزات - أمن البناء)
- أمن الأفراد والتشغيل (حماية الأفراد العاملين على المنظومات المعلوماتية وحماية تفاصيل بعض العمليات أو النشاطات)
- الاستمرارية في العمل ونظام التعافي من الكوارث *continuity & disaster recovery system* (النسخ الاحتياطي، وحدات التغذية عديمة الانقطاع، مولد الكهرباء، تكرار النظام المعلوماتي، سياسة أمن المعلومات للتعافي من الكوارث واستعادة العمل).

1 - 5 - إدارة أمن النظم المعلوماتية

إدارة الأمن *security management* في النظم المعلوماتية تتطرق إلى:

- ماهي الممتلكات الواجب حمايتها
- ماهي مهددات هذه الممتلكات
- ما يمكن عمله لمعاكسة هذه المهددات

يتم في إدارة الأمن في النظم المعلوماتية:

- وضع سياسة أمنية للنظام المعلوماتي
- تقدير المخاطر على النظام المعلوماتي
- تنفيذ السياسة الأمنية
- التفقد الدوري والصيانة للإجراءات الأمنية

ويمكن تعريف إدارة الأمن في الأنظمة المعلوماتية: بأنها العملية التي تقوم بإنجاز والحفاظ على مستويات مناسبة من السرية والسلامة والإتاحية والمساءلة والاستيقان والوثوقية.

a process used to achieve and maintain appropriate levels of confidentiality, integrity, availability, accountability, authenticity and reliability.

حيث ظهرت كلمة وثوقية لكي نتحقق من تنفيذ الأهداف الخمسة لأمن المعلومات والنظم المعلوماتية كما ينبغي بجودة عالية.

وقد وضعت عدد من المعايير الدولية بهدف إنشاء وصيانة نظام فعال لإدارة أمن المعلومات والنظم المعلوماتية. هناك سلسلة من المعايير العالمية المعتمدة لأمن نظم المعلومات *international security management standard (ISMS)* موصفة بـ *ISO27000 (ISO27001, ISO27002,... etc)* والتي تتضمن توصيات تتعلق بكافة مجالات أمن النظم المعلوماتية وإدارتها بدأً من وضع السياسة الأمنية وتقدير المخاطر وصولاً إلى أمن البيانات والشبكات والأمن الفيزيائي وأمن الأفراد وإدارة استمرارية العمل ونظام التعافي من الكوارث وانتهاءً بالحصول على شهادة الجودة بإدارة أمن النظم المعلوماتية *ISMS*.

الفصل الثاني

أدوات التعمية *Cryptographic Tools*

2 - *1* - تعاريف

التعمية *encryption* أو التشفير هي العملية التي يتم فيها تحويل نص واضح إلى نص آخر غير مفهوم *(بهدف إخفائه)* عن الأشخاص غير المخولين بالإطلاع عليه باستعمال طريقة محددة أو خوارزمية يستطيع من يعرفها أن يعيد استرجاع النص الواضح بعملية معاكسة تدعى فك التعمية *decryption* أو فك التشفير.

عرف العرب هذا العلم منذ القرن الثالث للهجرة *(التاسع الميلادي)* وأسموه علم التعمية. فالتعمية لغةً: الإخفاء والتلبيس، وهي في الاصطلاح: تحويل نص واضحٍ إلى آخر معمى *(غير مفهوم)*. باستعمال طريقة محددة، يستطيع من يعرفها أن يفهم النص. أما فك التعمية فيجري فيه تحويل النص المعمى إلى نص واضحٍ دون معرفة مسبقة لطريقة التعمية المستعملة. وقد عمل العرب أيضاً في استخراج المعمى وهو تحويل النص المعمى إلى نص واضح، من غير معرفة طريقة التعمية المستخدمة. فكلمة التعمية مكافئة لكلمة تشفير أينما وردت في النص.

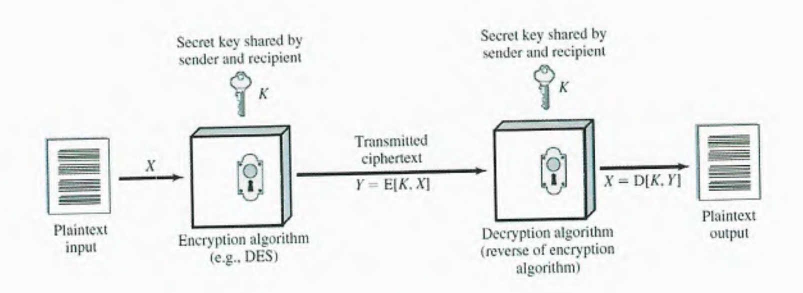
مصطلح التشفير *cryptography* يشير إلى عملية التشفير *encryption* وفك التشفير *decryption*. كما يشار أيضاً إلى عملية التشفير بالكلمة الإنكليزية *ciphering* وإلى عملية فك التشيفر *.deciphering*

أما استخراج المعمى أو ما يسمى تحليل الشيفرة *cryptanalysis* هي العملية التي يتم فيها استخراج المعمى أو الوصول للمعلومات الأصلية الواضحة دون معرفة خوارزمية التعمية أو المفتاح المستخدم في خوارزمية التعمية (كسر الشيفرة).

علم التعمية *cryptology* هو العلم الذي يضم كل من عمليتي التعمية أو التشفير *cryptography* وتحليل الشيفرة *cryptanalysis* استخراج المعمى.

المشفر *cipher* اصطلاحاً يقوم بعمليتي التشفير وفك التشفير. ويحكم عمله في عملية التشفير كل من خوارزمية التشفير ومفتاح تشفير آني متغير مع الزمن وفي عملية فك التشفير خوارزمية فك التشفير المعاكسة لخوارزمية التشفير ومفتاح فك التشفير.

مفتاح التشفير هو متحول سري، معروف فقط لطرفي الاتصال الذين يتبادلون رسالة محددة ويشكل دخلاً لخوارزمية التشفير. وتنتج خوارزمية التشفير قيماً مختلفة للنص المشفر بالاعتماد على مفتاح التشفير المستخدم.



(الشكل 1) الشكل العام لنظام تشفير (تعمية)

يبين الشكل *(1)* شكلاً عاماً لنظام تشفير أو تعمية. يسمى دخل نظام التشفير *(في طرف الإرسال)* وخرج نظام التشفير *(في طرف الاستقبال)* اصطلاحاً "نص واضح" *plaintext* وهو عبارة عن نص مكتوب بأي لغة. بينما يسمى خرج نظام التشفير *(المرسل عبر قناة الاتصال)* بـ "نص مشفر" *ciphertext*. ويتم استخدام مفتاح *key* للتشفير وفك التشفير.

لابد من الإشارة هنا إلى أن مصطلح "نص واضح" في نظام التعمية المبين في الشكل *(1)* يمكن أن يمثل نصاً مكتوباً بأي لغة، أو بصورة عامة بيانات رقمية تمثل نص أو صورة *(خرج كاميرا مرمز رقمياً* *digital camera* أو آلة فاكس أو إشارة تلفزيونية*)* أو صوت كلامي *(خرج جهاز صوتي أو هاتف مرمز رقمياً* *digital voice system)* أو بيانات حاسوبية *data* من الحاسوب او شبكة معلوماتية.

استخدامات التشفير

للتشفير تطبيقات واسعة في المجالات الدبلوماسية والعسكرية والأمنية والتجارية والإعلامية والمصرفية والمعلوماتية. يستخدم التشفير بشكل واسع:

- في المراسلات النصية المكتوبة بين الأفراد والشركات والبعثات الدبلوماسية والجهات الحكومية والدول بصورة عامة لحماية المعلومات ولمنع الاطلاع عليها خلال عملية نقلها.
- في أنظمة الاتصالات الشخصية للحفاظ على الخصوصية كما هو الحال في كافة أجهزة الهاتف الجوال *mobile* لمنع المتنصين على الاتصالات *(مسترقي السمع)* من فهم المحادثات الهاتفية.
- في أنظمة الاتصالات العسكرية والأمنية الخاصة للحفاظ على سرية المعلومات المنقولة سواء أكانت محادثات كلامية أو نصوص مكتوبة أو صور ومخططات مختلفة.

- في أنظمة الاتصالات التجارية للحفاظ على سرية التبادلات التجارية، وفي المصارف حفاظاً على سرية العمليات المالية والمصرفية. ويستخدم بشكل واسع في إشارات البث التلفزيوني الرقمي عبر الأقمار الصنعية بهدف الحماية التجارية ومنع غير المشتركين من فك تعمية الصورة التلفزيونية ورؤية البرامج.
- في الشبكات المعلوماتية التي تنقل المعلومات بين الحواسيب للحفاظ على خصوصية أو سرية البيانات والمعلومات المنقولة.
- في شبكة الإنترنت للحفاظ على خصوصية أو سرية البيانات والمعلومات المنقولة حسب التطبيق المستخدم. علماً بأن كل متصفحات الإنترنت المشهورة مثل *Internet Explorer* تتضمن مشفرات مبنية ضمنها لأغراض تطبيقات التجارة الإلكترونية الآمنة والدفع الإلكتروني بواسطة البطاقات الائتمانية.
- في النظم الحاسوبية لحفظ وتخزين المعلومات المصنفة بالأهمية العالية أو عالية السرية.
- في التوقيع الإلكتروني والتحقق من الشخصية وسلامة المعلومات.
- في إخفاء كلمات المرور الحاسوبية.

2 - 2 - أنواع التعمية

تتضمن طرق التشفير نوعين رئيسين هما:

1- طرق التشفير المتناظرة *symmetric encryption* (وتتضمن نوعين التسلسلية والكتلية)

2- طرق التشفير غير المتناظرة *asymmetric encryption* (وتتضمن خوارزميات التشفير بالمفتاح العمومي)

وسنتاولهما بمزيد من التفصيل في الفقرات اللاحقة.

يمكن كتابة الرسالة المشفرة C كتابع رياضي (بمثل خوارزمية التشفير) E لكل من الرسالة الواضحة M ومفتاح التشفير K_e كما يلي: $C= E(M, K_e)$

وعملية فك التشفير D كتابع لكل من الرسالة المشفرة C ومفتاح التشفير K_d كما يلي: $M= D(C, K_d)$

بتعويض C في العلاقة الأولى نجد: $(M= D(E(M, Ke), Kd$

فعندما تكون خوارزمية التشفير مماثلة لخوارزمية فك التشفير ومفتاح التشفير يساوي مفتاح فك التشفير ($D=E$ و$K_e = K_d$) فإن عملية التشفير تكون متناظرة ويدعى المفتاح في هذه الحالة بالمفتاح السري *secret key*. بينما في عملية التشفير غير المتناظرة يستخدم مفتاح تشفير مختلف عن مفتاح فك التشفير وربما خوارزمية تشفير مختلفة عن تلك المستخدمة في التشفير. ومن أشهر طرق التشفير غير المتناظرة تلك التي تستخدم التشفير بالمفتاح العمومي *public key encryption* حيث يتم استخدام مفتاح تشفير معلن بينما يكون مفتاح فك التشفير خاص أو سري.

2 - 3 - المشفرات التقليدية – التشفير بالمفتاح العشوائي الوحيد

تُصنف كافة طرق التشفير التقليدية القديمة ضمن المشفرات المتناظرة. ونذكر منها على سبيل المثال لا الحصر مايلي:

- مشفرات التبديل *Permutation Ciphers*
- التشفير بالمفتاح الوحيد *One time Pad Ciphers (OTP)*

وهذه المشفرات تستخدم لتشفير النصوص.

في مشفرات التبديل يتم استبدال حروف النص الواضح بحروف أخرى ,وفق تبديلة معينة لحروف اللغة. لدينا *28* حرف باللغة العربية وفق مايلي:

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28
أ	ب	ت	ث	ج	ح	خ	د	ذ	ر	ز	ع	غ	س	ش	ص	ض	ط	ظ	ف	ق	ك	ل	م	ن	هـ	و	ي

تستخدم تبديلة للحروف الـ *28* بطريقة غير منتظمة وفق مايلي:

ر	ل	و	ع	ز	ط	ن	غ	ظ	ق	م	ي	د	هـ	ج	ك	ت	ذ	خ	س	أ	ض	ش	ب	ف	ح	ث	ص

وكمثال على تشفير نص واضح وفق هذه الطريقة:

النص الواضح "عربه مصفحه"

النص المشفر "يقلح بكسطح"

ويكون لدينا في هذه الطريقة $26! > 2^{88}$ تبديلة مختلفة لمفتاح التشفير.

تعتبر هذه الطريقة سهلة الكسر باستخدام المعلومات الإحصائية للغة. حيث يستبدل حرف بآخر وبالتالي يحافظ النص المشفر على التوزع الإحصائي المعروف لحروف كل لغة ضمن أي نص طويل نسبياً. وهناك طرق أخرى أكثر تعقيداً. لكن من أهم وأفضل المشفرات التقليدية طريقة التشفير بمفتاح عشوائي وحيد.

التشفير بالمفتاح العشوائي الوحيد

للتخلص من التحليل الإحصائي للنص المشفر تستخدم طريقة التشفير بمفتاح تشفير عشوائي وحيد غير متكرر *one time pad (OTP)* غالباً مايكون موجود على ورقة *pad*. ولهذه الطريقة شرطان أساسيان هما:

- استخدام سلسلة من الحروف المولدة بشكل عشوائي (ذات توزع أقرب مايكون إلى التوزع المنتظم أي احتمال ظهور كل حرف في سلسة المفتاح العشوائي متساوي تقريباً لكل الحروف)
- عدد حروف مفتاح التشفير العشوائي يساوي إلى عدد حروف النص الواضح المراد تشفيره.
- يستخدم المفتاح العشوائي مرة واحدة في عملية التشفير ويتلف بعدها.

طريقة التشفير:

حيث يجمع ترتيب الحرف الواضح P ضمن الابجدية مع ترتيب حرف مفتاح التشفير العشوائي المقابل له K_e ويؤخذ الرقم الناتج $mod\ 28$ فنحصل على ترتيب الحرف المشفر.

$$P= (C+28-K_e)\ mod\ 28$$

وفك التشفير

تتميز هذه الطريقة عندما تستخدم وفق الشروط المذكورة أعلاه بما يلي:

- النص المشفر لايتضمن أي معلومات متبقية عن النص الواضح. والحرف الواحد الواضح يأخذ بعد التشفير كل حروف اللغة حسب المفتاح العشوائي المستخدم.
- كل حروف النص الواضح لها نفس احتمال تشفير متساوي.
- مفتاح التشفير معروف فقط من مرسل الرسالة والمستقبل لها ويوزع قبل إجراء الاتصال.

عند استخدام مشفر المفتاح العشوائي الوحيد *OTP* في الحاسوب يكون لديه إمكانية لطريقة تشفير أكثر قوة. يمثل أي ملف حاسوبي بسلسلة من البايتات *bytes* كل منها تتضمن ثماني بيتات *8 bits* بغض النظر عن محتوى الملف سواء أكان نص أم صورة أم رسومات أم صوت. ويؤخذ المفتاح العشوائي على شكل سلسلة من البايتات العشوائية. وتتم عملية التشفير هنا بالجمع الثنائي بدون حامل *XOR* بين بايتات الملف الواحدة تلو الأخرى وبايتات المفتاح العشوائي الواحدة تلو الأخرى وكمثال على ذلك مايلي:

في عملية التشفير في طرف الإرسال:

بايت واضحة P ممثلة بالترميز الثنائي **01011001**

بايت من مفتاح التشفير العشوائي K_e **01010101**

ناتج عملية التشفير C (جمع ثنائي دون حامل) بايت مشفرة **00001100**

في عملية فك التشفير في طرف الاستقبال:

بايت مشفرة C **00001100**

نفس البايت K_e من مفتاح التشفير العشوائي **01010101**

البايت الناتجة بعد فك التشفير هي نفس P الواضحة **01011001**

طبعاً لابد من التنويه أن مفتاح التشفير العشوائي (السري) مولد مسبقاً وموزع لكلا طرفي الإرسال-تشفير والاستقبال -فك التشفير. وهذه هي إحدى مساوئ التشفير بمفتاح سري المتمثلة في صعوبة توزيع المفاتيح قبل بدء عملية التشفير. يستخدم هذا النوع من التشفير على نطاق محدود في التطبيقات الدبلوماسية والمراسلات الاستراتيجية ولرسائل قصيرة نسبياً تعد وتشفر مسبقاً كونه من الصعوبة بمكان توليد مفاتيح عشوائية طويلة بالزمن الحقيقي.

لكن هذا النوع من المشفرات منيع ضد الكسر عند تحقق الشروط المذكورة أعلاه. وبصورة عامة تكون المشفرات آمنة من الناحية العملية الحسابية عندما يكون:

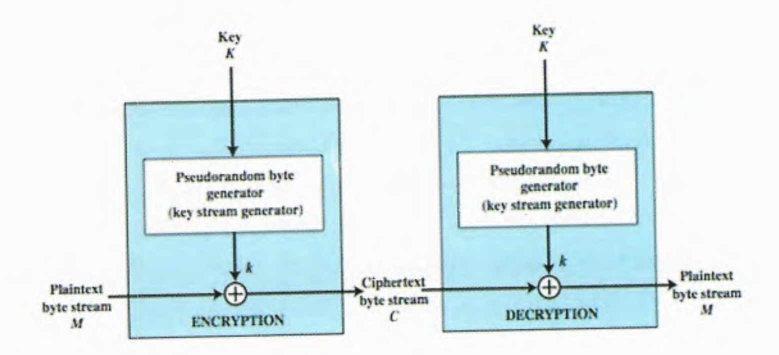
- كلفة كسر المشفر تزيد عن كلفة قيمة المعلومات المشفرة
- الزمن اللازم لعملية كسر الشيفرة يفوق الزمن المفيد لبقاء المعلومات سرية

من الصعب تقدير الجهود اللازمة لكسر المشفرات حيث نفترض أن المهاجم يعلم خوارزمية التشفير ويحتاج لمفتاح التشفير فقط لفك الشيفرة. حيث يمكن استخدام طريقة البحث الشامل (هجوم القوة الغاشمة) *brute force attack* التي تأخذ بعين الاعتبار تجريب كل المفاتيح الممكنة لفك الرسالة المشفرة حتى نصل للرسالة الأصلية المتماسكة. لأنه عملياً مفاتيح غير صحيحة قد تعطي رسائل إيجابية خاطئة. فمثلاً لرسالة نصية (تستخدم *26* حرف من حروف الأبجدية) مشفرة مكونة من *1000* حرف (صفحة واحدة) هناك 26^{1000} إمكانية لمفتاح التشفير العشوائي المستخدم. وإذا كان التشفير حاسوبي باستخدام مفتاح عشوائي مكون من سلسلة بايتات فإننا نحتاج إلى تجريب 256^{1000} مفتاح. وبالتالي من المستحيل تجريب كل الحالات التي يأخذها المفتاح.

2 - 4 - المشفرات المتناظرة التسلسلية

تستخدم المشفرات التسلسلية *stream ciphers* كبديل عملي عن مشفرات المفتاح العشوائي الوحيد. يمثل الشكل *(2)* المخطط العام للمشفرات التسلسلية حيث يتم التشفير من خلال عملية *XOR* (جمع ثنائي بدون حامل) بين بايتات النص الواضح وبين سلسلة بايتات مولدة عبر مولد مفتاح شبه عشوائي *pseudo random generator (PRG)* طوله غير محدود يولد من خلال خوارزمية تُقاد بمفتاح سري خارجي

محدود الطول (مثلاً 64 بت). هذه الحالة مماثلة لحالة التشفير بالمفتاح العشوائي الوحيد لكن هنا يتم توليد سلسلة المفتاح شبه العشوائي باستخدام خوارزمية قابلة للإعادة وتولد بطول غير محدود من بايتات سلسلة المفتاح. طول سلسلة المفتاح شبه العشوائي وتعقيدها الرياضي وطول المفتاح السري الخارجي الذي يقودها يحدد قوة هذا النوع من المشفرات. وغالباً مايكون طول السلسلة مساوي إلى $2N$ حيث N تمثل عدد بيتات المفتاح الخارجي. وتستخدم خوارزميات التشفير التسلسلية عمليات منطقية بسيطة مثل الجمع الثنائي XOR والإزاحة والاستبدال.



(الشكل 2) الشكل العام لمشفر تسلسلي

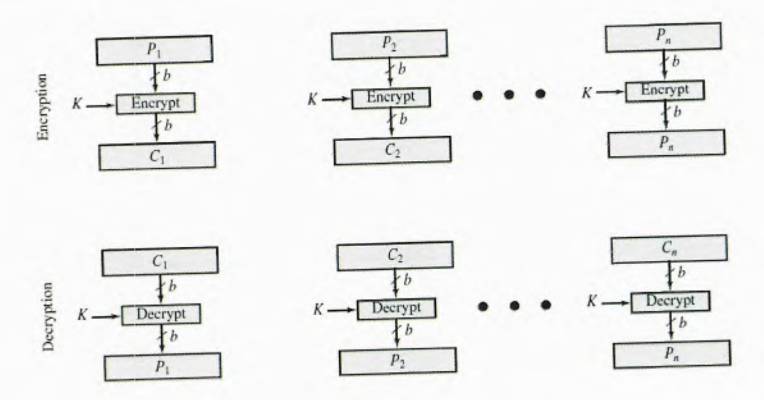
في هذا النوع من المشفرات المتناظرة يجب تبادل مفاتيح التشفير السرية الخارجية قبل بدء التشفير وغالباً مايتم تغيير هذه المفاتيح بشكل يومي لنحصل على سلسلة جديدة من بايتات المفتاح شبه العشوائي المستخدم في التشفير.

تتميز خوارزميات التشفير التسلسلية بالسرعة العالية وتستخدم في تجهيزات الاتصالات المحكية كونها لاتسبب أي تأخير زمني نتيجة عملية التشفير. ومن أشهر هذه الخوارزميات خوارزمية *A5* والتي تمثل **مولد بيتات شبه عشوائي** *bit stream generator*. وتستخدم في تشفير قناة الصوت الرقمية في جهاز الموبايل وخوارزمية *RC4* والتي تمثل **مولد بايت شبه عشوائي** *byte stream generator* وتستخدم في بروتوكولات أمن الشبكات اللاسلكية المبنية على تقانة *IEEE 802.11 (WiFi)* مثل الشبكات المحلية اللاسلكية *WLAN: Wireless Local Area Networks*.

2 - 5 - المشفرات المتناظرة الكتلية

في المشفرات الكتلية *block ciphers* يتم التعامل مع كتلة *N* من البيتات على عكس المشفرات التسلسلية التي تتعامل مع بايت أو بت الواحدة تلو الأخرى من النص الواضح.

يتطلب هذا النمط من خوارزميات التشفير تقسيم النص الواضح *plaintext* إلى كتل متساوية الطول *n-bit* $(P_1, P_2, ...P_n)$ مع إتمام الكتلة الأخيرة بالأصفار وصولاً إلى الطول المطلوب، ومن ثم تعمية كل كتلة على حدى باستخدام الخوارزمية ذاتها والمفتاح k ذاته، وفي النهاية تشكيل النص المشفر *ciphertext* من تجميع الكتل المشفرة $(C_1, C_2,.. C_n)$. أما فك التشفير *decryption* فيتم بتنفيذ العمليات المعاكسة بشكل كتلي باستخدام خوارزمية فك التشفير ونفس مفتاح التشفير k وفق ماهو مبين في الشكل *(3)*.



(الشكل 3) الشكل العام للمشفر الكتلي

يتوفر العديد من خوارزميات التعمية المتناظرة وأشهرها:

- خوارزميات *data encryption standard (DES)* المعيارية والتي تتميز بطول الكتلة *64-bit* وطول مفتاح التشفير *56-bit* حيث اعتمدت كمعيار عام *1977* من قبل المعهد الوطني الأمريكي للمعايير والتقانة *NIST: National Institute for Standards and Technology*.

- خوارزمية *advanced encryption standard (AES)* المعيارية والتي تدعى أيضاً باسم *Rijndael* وتم اعتمادها من معهد *NIST* عام *2001* بعد وضعها في الاختبارات لمدة خمس سنوات تتميز بطول كتلة *128-bit* ولها ثلاثة أنواع حسب طول مفتاح التشفير *AES-128, AES-192, AES-256*، *128* أو *192* أو *256* بت على الترتيب.

- خوارزمية *Triple DES* أو *3DES* وتم اعتمادها كمعيار من معهد *NIST* في عام *1999* وذلك نتيجة المنافسة مع خوارزمية *AES* ذات المفتاح الأطول (والتي كانت قيد الاختبارات). في خوارزمية *3DES*

يتم تطبيق خوارزمية *DES* ثلاث مرات متتالية باستخدام ثلاثة مفاتيح كل منها بطول *56* بت بطول إجمالي *168-bit* وفق العلاقة التالية:

$$C = E(E(E(P,K1),K2),K3)$$

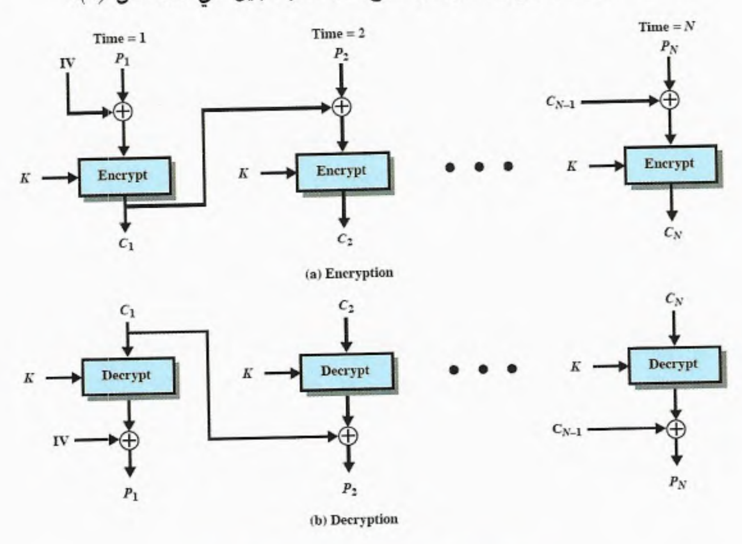
كما أن نسخة من الخوارزمية تستخدم مفتاحين مختلفين *k1* و *k2* كل منهما بطول *56-bit* ليصبح طول المفتاح *112-bit* لكنه يؤدي التشفير بمفتاح *168-bit* وفق العلاقة التالية:

$$C = E(D(E(P,K1),K2),K1)$$

فك التشفير *D* (للكتلة المشفرة بمفتاح *k1*) باستخدام المفتاح *k2* هو تشفير. والهدف من ذلك أنه عندما يتساوى المفتاحان *k1* و*k2* تصبح *3DES* مطابقة لخوارزمية *DES*. في الحياة العملية عندما يتم الاتصال بين موجهين *routers* أحدهما يمتلك خوارزمية *3DES* بينما الأخر يمتلك فقط خوارزمية *DES*، يتم خلال عملية المصافحة بين الموجهين ببداية أي جلسة جديدة للاتصال اعتماد *DES* للتشفير في كليهما ويسخدم الموجه الاول *k1=k2* عندها تصبح *3DES* مطابقة لـ *DES*.

إن وجود كتلتين *P* متطابقتين في النص الواضح يعطي كتلتين متطابقتين في خرج المشفر الكتلي. وغالباً ما تفيد هذه المعلومة محللي الشيفرة في عمليات كسر التشفير. ولتلافي نقطة الضعف هذه تُستخدم عدة طرق أشهرها ما يسمى بطريقة سلسلة الكتل المشفرة *cipher block chaining (CBC)* والتي تقوم بإخفاء النص الواضح قبل تقديمه للمشفر من خلال استخدام كتلة ابتدائية *initial vector (IV)* بطول كتلة النص المراد تشفيره وإجراء عملية الجمع الثنائي بدون حامل *XOR* بين الكتلة الابتدائية والكتلة الأولى

من النص الواضح. بينما تستخدم الكتلة المشفرة الأولى الناتجة كقيمة ابتدائية للكتلة الثانية وهكذا لباقي الكتل كما يبين الشكل *(4)*. وتجري العمليات العكسية في فك التعمية للحصول على الكتل الأصلية واسترجاع النص الواضح كما هو مبين في الشكل *(4)*.



الشكل *(4)* سلسلة الكتل المشفرة *CBC*

عملياً يتراوح طول مفتاح التشفير الخارجي *(السري)* في المشفرات المتناظرة من *56-256* بت. ففي خوارمية *DES* يكون طول المفتاح *56* بت بينما يصل إلى *112* بت في *3DES* و*256* بت في خوارزمية *AES*. وتزيد مناعة التشفير كلما كان المفتاح أطول مما يزيد من صعوبة تنفيذ هجوم البحث الشامل من الناحية العملية بحيث يصبح من الصعوبة تجريب كافة القيم المحتملة لمفتاح التشفير في فك نص مشفر بخوارزمية معينة حتى الحصول على نص مقروء وصحيح

وبالتالي على مفتاح التشفير. يبين الجدول التالي الوقت المطلوب لتنفيذ هجوم في حالات مختلفة ويثبت استحالة نجاحه من الناحية العملية من أجل أطوال مفاتيح كبيرة:

Key size (bits)	Cipher	Number of Alternative Keys	Time Required at 10^9 decryptions/μs	Time Required at 10^{13} decryptions/μs
56	DES	$2^{56} \approx 7.2 \times 10^{16}$	$2^{55} \mu s = 1.125$ years	1 hour
128	AES	$2^{128} \approx 3.4 \times 10^{38}$	$2^{127} \mu s = 5.3 \times 10^{21}$ years	5.3×10^{17} years
168	Triple DES	$2^{168} \approx 3.7 \times 10^{50}$	$2^{167} \mu s = 5.8 \times 10^{33}$ years	5.8×10^{29} years
192	AES	$2^{192} \approx 6.3 \times 10^{57}$	$2^{191} \mu s = 9.8 \times 10^{40}$ years	9.8×10^{36} years
256	AES	$2^{256} \approx 1.2 \times 10^{77}$	$2^{255} \mu s = 1.8 \times 10^{60}$ years	1.8×10^{56} years

بالنتيجة تتميز المشفرات المتناظرة بمايلي:

- **السرعة العالية:** كونها تقوم بعمليات بسيطة مثل الجمع *xor* والإزاحة والاستبدال وبالتالي قابلة للتنفيذ حاسوبياً بالزمن الحقيقي
- **مفتاح التشفير الخارجي:** مفتاح سري واحد للتشفير وفك التشفير
- **طول مفتاح التشفير:** يتراوح مابين *56-256* بت
- **توزيع المفاتيح:** يجب أن يمتلك كل من المرسل والمستقبل مفتاح التشفير الخارجي قبل بدء عملية التشفير.

من النقطة الأخيرة نجد أن المشفرات المتناظرة تعاني من مشكلة توزيع وتبادل المفاتيح قبل بدء عملية التشفير.

2 - 6 - إدارة مفاتيح التشفير

من أهم الترتيبات المتعلقة بالتشفير هي كيفية إدارة وتوزيع مفاتيح التشفير. فعلى الرغم من إمكانية توفير نظم وخوارزميات تشفير منيعة فإن سوء الاستخدام وعدم الإدارة الصحيحة لمنظومات التشفير ومفاتيحها يلغى فاعليتها. فإدارة مفاتيح التشفير ونظم التشفير من الأمور الهامة في منظومات التشفير وتصل أهميتها إلى أهمية مناعة خوارزمية التشفير. وتعاني طرق التشفير المتناظرة من مشكلة توزيع مفاتيح التشفير الخارجية السرية قبل التشفير. وتظهر هذه المشكلة بشكل واضح عندما يكون لدينا شبكة ذات عدد كبير من محطات الاتصال والتي تحتاج لتوليد مئات المفاتيح الخارجية وتحديد طريقة تعاملها وتوزيعها بشكل مسبق. وسابقاً كان يستخدم محطة أو مركز خاص لتوليد وتوزيع المفاتيح تُعرف فيه شبكة الاتصال ويقوم بتوليد مفاتيح التشفير.

من هنا كان هناك حاجة لإيجاد طريقة بديلة يتم فيها تبادل مفاتيح التشفير الخارجية السرية بطريقة آمنة قبل البدء بعملية التشفير. ومن أشهر هذه الطرق طريقة التشفير بالمفتاح العمومي العلني.

2 - 7 - خوارزميات التشفير بالمفتاح العمومي

تعتبر خوارزميات التشفير بالمفتاح العمومي *public key cryptography* من الخوارزميات غير المتناظرة. كونها تسخدم مفتاح تشفير مختلف عن مفتاح فك التشفير. يتم في طريقة التشفير بالمفتاح العمومي العلني استخدام زوج من المفاتيح أحدهما معلن *public key* ويستخدم للتشفير ويسمى e والآخر سري أو خاص *private key* ويستخدم لفك التشفير ويسمى d. وهذان المفتاحان مترابطان رياضياً بحيث لايمكن فك الرسالة المشفرة بطريقة التشفير بخوارزمية المفتاح العمومي باستخدام المفتاح e إلا باستخدام المفتاح d. فإذا كان

لدينا طرفا اتصال A و B يرغبان بالاتصال المشفر مع بعضهما فإن الطرف A يولد زوج مفاتيح e_a و da والطرف B يولد زوج مفاتيح e_b و d_b، وقبل بدء الاتصال يتبادلان مفاتيح التشفير العمومية لكل منهما أو تكون مفاتيح التشفير العمومية ea و e_b معلنة ضمن دليل مفاتيح عمومية معلنة على شبكة الإنترنت.

في مشفر المفتاح العمومي، يقوم الطرف A بتشفير الرسائل المرسلة إلى الطرف B باستخدام خوارزمية التشفير بالمفتاح العمومي التي تستخدم المفتاح العمومي e_b للطرف B بينما يقوم الطرف B بفك تشفير الرسالة المشفرة المستلمة باستخدام خوارزمية فك التشفير التي تستخدم مفتاحه الخاص d_b. وبالتالي هو الوحيد القادر على فك الرسالة المشفرة كونه الوحيد الذي يمتلك مفتاح فك التشفير d_b.

تمكننا طريقة التشفير بالمفتاح العمومي من التحقق من هوية مصدر البيانات *data source authentication*. فإذا قام المرسل بتشفير الرسالة بخوارزمية التشفير بالمفتاح العمومي باستخدام مفتاحه الخاص فسيتمكن كل من يستلم الرسالة من فك تشفير الرسالة كون المفتاح العمومي معروف ومعلن. وعند نجاح عملية الفك سيتم التأكد من هوية المرسل كون المرسل هو الوحيد الذي يمتلك مفتاحه الخاص.

لتلافي الخداع في انتحال شخصية ونشر مفتاح عمومي لها، لابد من الربط بين المفتاح العمومي وهوية صاحب المفتاح، ويتم ذلك من خلال شهادة رقمية *digital certificate* تسمى أيضاً شهادة المفتاح العمومي *public key certificate* يصدرها طرف ثالث يسمى سلطة إصدار الشهادة *certificate authority (CA)* تتضمن بيانات صاحب المفتاح العمومي والمفتاح العمومي وخوارزميتي الضغط وتشفير مختصر الرسالة وفترة صلاحية الشهادة. وتتضمن أيضاً التوقيع

الالكتروني للجهة التي أنشأتها CA. ويطلق في القانون السوري اسم "مزود خدمات التوقيع الألكتروني" على سلطة إصدار الشهادة CA. وسيتم التطرق لها وشرحها بمزيد من التفصيل في فقرة التوقيع الرقمي.

خوارزمية RSA

من أشهر خوارزميات التشفير بالمفتاح العمومي خوارزمية RSA المنشورة عام 1978. والتي تتضمن ثلاث خطوات رئيسية هي: توليد زوج المفاتيح، التشفير، فك التشفير. فمثلاً لرسالة M توصف الخوارزمية بالعلاقات التالية:

في التشفير $C = M^e \bmod N \text{ where } M < N$

في فك التشفير $M = C^d \bmod N = M^{ed} \bmod N$

ويتحقق الفك عندما يكون جداء المفتاحين ed يساوي 1 وفق العلاقة $ed = 1 \bmod \Phi(n)$

حيث التابع $\Phi(n)$ هو Euler totient function (نظرية الأعداد).

يمكن البرهان على أنه من أجل عددين أوليين p و q لدينا

$N = p.\, q$ ولدينا $\Phi(n) = (p-1)(q-1)$

يتم انتقاء المفتاح e بحيث يحقق العلاقة:

$$gcd(\Phi(n), e) = 1; \text{ where } 1 < e < \Phi(n)$$

حيث يمثل gcd القاسم المشترك الاعظمي، وبالتالي المفتاح العمومي مكون من $\{ e , N \}$

يتم حساب d بحيث يحقق العلاقة:

$$ed = 1 \bmod \Phi(n)$$

ويكون المفتاح الخاص $\{d, N\}$. وتبقى المتحولات التالية $(p, q, d, \Phi(n))$ سرية غير معلنة.

في الحالات العملية تكون الأرقام الأولية p و q من رتبة 155 رقم عشري أي 512 bits وبالتالي معرفة N يجعل من الصعب تحليلها إلى أرقامها الاولية p و q. وغالباً ماتكون هذه الارقام ضمن المجال 512-2048 bits أي مابين 155-617 رقم عشري على الترتيب.

يقدر أنه يلزم 10^{17} MIPS-year لتحليل رقم مكون من 2048 bits إلى عوامله الاولية.

والمقياس MIPS-year يمثل معالج حاسوبي يقوم بمليون تعليمة في الثانية ولمدة سنة (أي $3.15\ 10^{13}$ عملية).

مثال على التشفير بالمفتاح العمومي باستخدام أرقام صغيرة للتوضيح:

بفرض أن $p=11, q=17$ فإن $N=187$ و $\Phi(n)= 160$

اختيار $e=7$ فإن $gcd(e, 160)=1$

وتكون $de = 1 \bmod \Phi(n) = 1 \bmod 160$

بتعويض قيمة e $\bmod 160 = 1$ فإن $23 * 7 = 161$ $d = 23$

من أجل رسالة مكونة من بايت ذات قيمة $M=88$ مثلاً فإن:

$$C = 887 \bmod 187 = 11$$

$$M = 1123 \bmod 187 = 88$$

بروتوكول ديفي- هيلمان *Diffie-Hellman*

تقدم خوارزمية ديفي-هيلمان *Diffie-Hellman* خدمة تشارك المفاتيح السرية، وهي من خوارزميات التعمية اللامتناظرة. وناتجها هو قيمة سرية مشتركة بين طرفين يمكنهما استخدامها كمفتاح سري في التعمية المتناظرة. تعمل خوارزمية ديفي-هيلمان اعتماداً على التخاطب بين طرفين، ولذلك تسمى بروتوكول. يهدف هذا التخاطب لتشارك قيمة سرية على الرغم من كون الاتصال بين الطرفين مكشوفاً، وذلك على الشكل التالي:

1. لنسمي المستخدم الأول *A(Alice)* والمستخدم الثاني *B(Bob)*.

2. لنفرض أن المستخدمين يتشاركان قيمتين، الأولى نسميها q وهي عدد أولي ضخم، والثانية نسميها g وهي عدد أصغر من q وجذر أولي *primitive root* له.

3. يقوم كل من A وB بتوليد قيمة عشوائية – a وb على التوالي – وتعتبر بالنسبة لكل منهما مفتاحه الخاص *private key* ويشترط بهذه القيمة أن تكون أصغر من q.

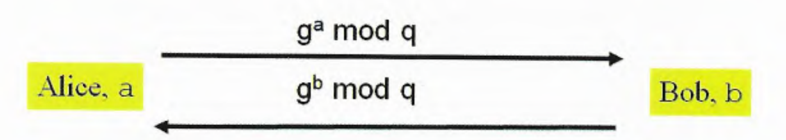
4. يقوم A بحساب مفتاحه العام *public key* بالعلاقة $X = g^a \ mod \ q$، ويرسله إلى B.

5. يقوم B بحساب مفتاحه العام *public key* بالعلاقة $Y = g^b \ mod \ q$، ويرسله إلى A.

6. يقوم A بحساب القيمة $K_1 = Y^a \ mod \ q$.

7. يقوم B بحساب القيمة $K_2 = X^b \ mod \ q$.

8. نلاحظ أن القيمة $K = K_1 = K_2 = g^{ab} \ mod \ q$ قيمة سرية مشتركة يمكن استخدامها لاحقاً كمفتاح سري في التعمية المتناظرة انظر الشكل (5).



شكل *(5)* مبدأ بروتوكول ***Diffie-Hellman*** لتشارك المفاتيح السرية

ملاحظات:

- تعتبر عملية حساب المفتاح العمومي انطلاقاً من الخاص لدى كل من الطرفين، وهي باقي القسمة على عدد أولي ضخم لنتيجة الرفع لقوة، عملية حساب سريعة نسبياً كمعالجة حاسوبية، بينما تعتبر العملية العكسية، وهي باقي القسمة على عدد أولي ضخم لنتيجة لوغاريتم، عملية صعبة رياضياً أو مستحيلة حسابياً، وتسمى اللوغاريتم المتقطع *discrete logarithm*.
- لن يتمكن مهاجم يتجسس على الخط المكشوف بين A وB من حساب المفتاح الخاص لأي من الطرفين انطلاقاً من المفتاح العمومي المقابل الذي ينتقل على خط الاتصال، حتى لو حصل هذا المهاجم على القيمتين q وg باعتبارهما ليستا سريتين، وذلك لأنه سيحتاج لحساب لوغاريتم متقطع.
- تعاني خوارزمية ديفي-هيلمان بنسختها الأولية المشروحة أعلاه من نقطة ضعف ناجمة عن عدم وجود تحقق من هوية المستخدم *user authentication*، ما قد يؤدي إلى تعرضها لاختراقات من نمط الرجل-في-المنتصف *man-in-the-middle attack* كما في الشكل *(6)* الذي يبين رجل المنتصف *Trudy*. ينتحل المهاجم في هذا الاختراق هوية كل من الطرفين أمام الآخر، وينشأ بالنتيجة مفتاح سري بين المهاجم وكل من الطرفين، ويظن كل منهما أن لديه مفتاح سري مع الطرف الآخر ويستخدمه في هذا الإطار.

يستطيع المخترق بالنتيجة التجسس على مراسلات الطرفين أو التعديل عليها أو تعطيلها.



الشكل *(6)* اختراق الرجل-في-المنتصف

- يتم التغلب على اختراق الرجل في المنتصف من خلال استخدام خوارزمية التشفير بالفتاح العمومي *RSA* حيث يقوم كل طرف بإرسال المفتاح السري للطرف الأخير بعد تشفيره بخوارزمية التشفير بالمفتاح العمومي باستخدام المفتاح العمومي للطرف الآخر. ويقوم كل طرف بفك التشفير بالمفتاح الخاص به مما يؤدي للتحقق من هوية الطرفين.

2 - 8 - مقارنة بين خوارزميات التشفير المتناظرة والتشفير بالمفتاح العمومي

يبين الجدول التالي مقارنة بين خوارزميات التشفير المتناظرة بالمفتاح السري وخوارزميات التشفير بالمفتاح العمومي

الواصفة	خوارزميات التشفير المتناظرة	التشفير بالمفتاح العمومي
السرعة	سريعة كونها تستخدم عمليات بسيطة جمع وإزاحة واستبدال	بطيئة لانها تستخدم عملية الرفع إلى قوة
عدد المفاتيح	مفتاح سري واحد للتشفير وفك التشفير	مفتاحين أحدهما معلن والأخر خاص سري
توزيع المفاتيح	يجب أن يكون المفتاح السري لدى طرفي الاتصال قبل البدء بالاتصال	مفتاح التشفير معلن
عدد بيتات مفتاح التشفير الخارجي	*56-256* بت *256* يعتبر آمن	*512-2024* بت مفتاح اطول
مثال	*DES, AES, A5 / 1*	*Diffie-Hellman* و *RSA*

في الحياة العملية تسخدم خوارزمية التشفير المتناظرة لتشفير الاتصال بينما تسخدم خوارزمية التشفير بالمفتاح العمومي لتبادل المفاتيح بين طرفي الاتصال في بداية كل جلسة اتصال. فمثلاً في موجهات الاتصال وفي بداية كل جلسة اتصال يتم تشكيل المفتاح السري من تبادل أجزاء المفاتيح باستخدام خوارزمية ديفي- هيلمان واستخدام خوارزمية التشفير بالمفتاح العمومي *RSA* لضمان هوية الطرفين (طبعاً مع وجود شهادات رقمية معتمدة من *CA* لضمان الربط بين هوية كل طرف ومفتاحه العام) وبعد تبادل المفاتيح السرية بين الطرفين يتم تشكيل المفتاح السري لاستخدامه في التشفير بخوارزمية متناظرة *3DES* أو *AES* أو غيرها كونها قابلة للتنفيذ بالزمن الحقيقي.

وكمثال آخر على استخدام كل من خوارزمية التشفير المتناظرة والتشفير بالمفتاح العمومي مايسمى بالظرف الرقمي.

الظرف الرقمي *Digital Envelope*

توفر تقنية الظرف الرقمي للطرفين المتراسلين إمكانية تحقيق سرية الرسالة بتشفيرها بخوارزمية تشفير متناظرة سريعة بالمقارنة مع التشفير بخوارزمية غير متناظرة، ولكن دون وجود حاجة لتشارك المفتاح السري بشكل مسبق، حيث توفر خوارزمية التشفير اللامتناظرة إمكانية نقل هذا المفتاح السري، وهو مفتاح الجلسة، من المرسل إلى المرسل إليه بشكل آمن يمنع تسريبه، مع القدرة على استخدام مفتاح جلسة مختلف مع كل رسالة إن تطلب الأمر. الجدير بالذكر أن تشفير مفتاح الجلسة باستخدام التشفير بالمفتاح العمومي لا يطرح إشكالية بطء المعالجة لكون مفتاح الجلسة محدود الطول.

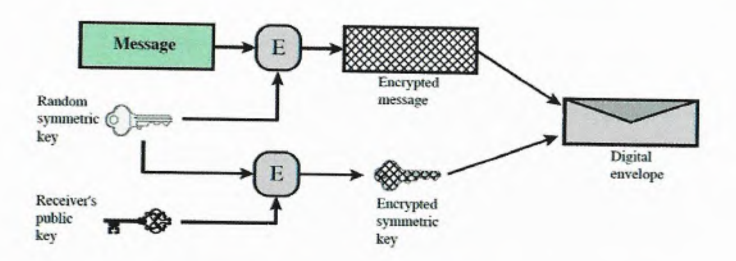
يتم تشكيل الظرف الرقمي كما يلي:

1. تحضير الرسالة

2. تشفير الرسالة باستخدام خوارزمية تشفير متناظرة E باستخدام مفتاح سري عشوائي *(للتشفير المتناظر)* يستخدم لمرة واحدة خلال جلسة تراسل واحدة، ويمكن تسميته بمفتاح الجلسة *session key*.

3. تشفير مفتاح الجلسة العشوائي باستخدام خوارزمية تشفير بالمفتاح العمومي باستخدام المفتاح العمومي للمرسل إليه. انظر الشكل *(7)*.

4. ربط مفتاح الجلسة المشفر مع الرسالة المشفرة لتشكيل الظرف الرقمي المطلوب إرساله.



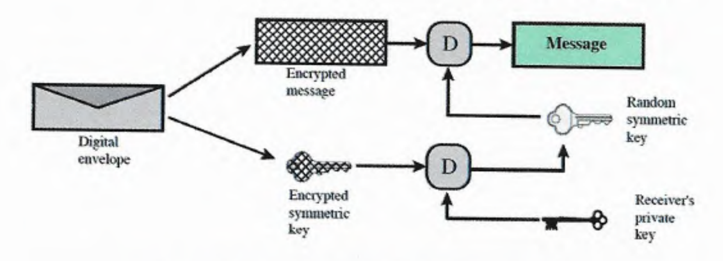
الشكل *(7)* طريقة إنشاء الظرف الرقمي

فقط الطرف المرسل إليه هو الوحيد القادر على فك شيفرة مفتاح الجلسة السري وبالتالي تمكنه من فك شيفرة الرسالة. طبعاً المرسل يحصل على المفتاح العمومي للطرف المرسل إليه من خلال شهادته الرقمية ليكون متأكداً من صحته وارتباطه بهوية الطرف المرسل إليه.

ويتم فك الظرف الرقمي لدى المرسل إليه كما يلي:

1. فك تشفير مفتاح الجلسة السري العشوائي المشفر باستخدام خوارزمية التشفير بالمفتاح العمومي وباستخدام مفتاحه الخاص أنظر الشكل *(8)*.

2. استخدام مفتاح الجلسة السري العشوائي الناتج في خوارزمية التشفير المتناظرة لفك تشفير الرسالة.



الشكل *(8)* طريقة فك تشفير الرسالة في الظرف الرقمي

2 - 9 - خوارزميات التحقق من الرسائل

المقصود بخدمة التحقق من الرسائل *message authentication* أن تسمح منظومة الأمن للمرسل إليه بالتحقق مما يلي:

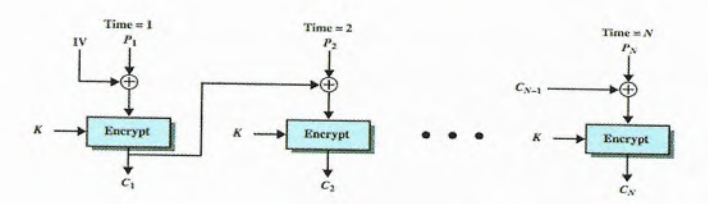
1. سلامة محتوى الرسالة *message integrity*، أي أنها وصلت إلى المرسل إليه دون أن تتعرض للتعديل.

2. صحة هوية مصدر الرسالة *message source authentication*، أي صحة هوية المرسل كما تشير إليها الرسالة.

3. وصول الرسالة في الوقت الصحيح وبالترتيب الصحيح (إذا تضمنت الرسالة ترقيم تتابعي)

خوارزميات التحقق من الرسالة *MAC*

خوارزمية كود التحقق من الرسالة *MAC: Message Authentication Code* هي تابع يربط بين الرسالة ومفتاح سري لإنتاج قيمة ثابتة الطول تستخدم للتحقق من صحة الرسالة وتسمى *MAC* أو مختصر الرسالة أو بصمة الرسالة كونها قيمة وحيدة تميز الرسالة عن غيرها. يحستب *MAC* بطريقة سلسلة الكتل المشفرة *cipher block chaining (CBC)* باستخدام مفتاح سري مشترك بين المرسل والمستقبل بحيث نحتفظ بأخر كتلة مشفرة من الرسالة C_N والتي تمثل *MAC* كما هو مبين في الشكل *(9)* والتي تضاف للرسالة. تفيد قيمة اللصاقة بحد ذاتها في التحقق من سلامة الرسالة لدى المرسل إليه، بينما يفيد استخدام المفتاح السري المشترك في التحقق من هوية المرسل.



CBC Final Block = MAC

الشكل *(9)* احتساب *MAC* وإضافته للرسالة

في طرف الاستقبال يتم احتساب *MAC* من الرسالة باستخدام خوارزمية *CBC* والمفتاح السري ومقارنتها بالكود *MAC* الواصل مع الرسالة وفي حال التطابق يتحقق من سلامة بيانات الرسالة ويتحقق من هوية المرسل *(*الطرف الذي يمتلك المفتاح السري*)*.

توابع التهشير ***Hash Functions***

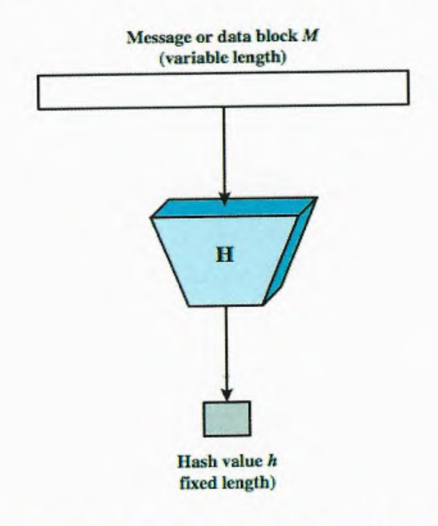
يصلح تابع التهشير *hash function* لتوليد مختصر للرسلة *message digest MD* أو بصمة للرسالة *(*قيمة ثابتة الطول تسخدم لضمان سلامة الرسالة*)* أو قيمة التهشير *hash value*، ولكنه لا يحتاج لمفتاح سري مع الرسالة في الدخل كما في حالة *MAC*. يجب أن يحقق تابع التهشير الخصائص التالية:

1. مهما كان طول الرسالة تكون قيمة التهشير ثابتة الطول وصغيرة نسبياً.

2. لا يمكن استرجاع الرسالة حسابياً انطلاقاً من قيمة تهشير، ولا حتى معرفة طول رسالة الدخل *(*تابع وحيد الاتجاه غير قابل للعكس*)*.

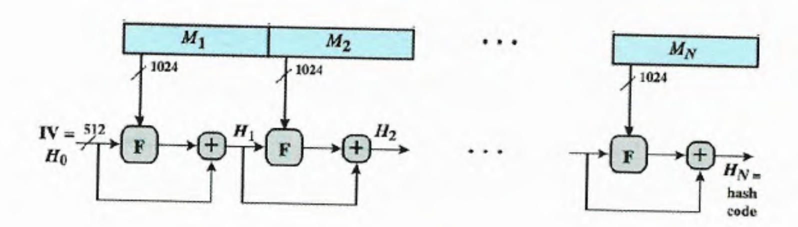
3. يجب أن يكون من غير الممكن حسابياً *computationally infeasible* استبدال رسالة دخل بواحدة أخرى تعطي في الخرج نفس البصمة من أجل نفس تابع التهشير.

4. يجب أن يكون من غير الممكن حسابياً إيجاد رسالتين لا على التعيين تعطيان في الخرج البصمة ذاتها من أجل تابع التهشير ذاته أي مقاومة شديدة للتصادم *strong collision resistance*



الشكل *(10)* تابع التهشير لكتلة واحدة من الرسالة

يبين الشكل *(11)* مخطط صندوقي عام لحساب تابع التهشير لرسالة مكونة من *N* كتلة كل منها *1024* بت والذي يتضمن *N* جولة وينتج عنه قيمة تهشير نهائية مساوية إلى *512* بت. يتم استخدام قيمة ابتدائية *initial value 512* بت في الجولة الأولى تُجمع مع قيمة التهشير الناتجة عنها *H1* ويُضغط الكتلة الأولى من الرسالة باستخدام تابع ضغط *compression function F*. وتتكرر نفس العمليات في كل الجولات *N*. التابع *F* مكون من عمليات منطقية بسيطة ما يجعل تميز توابع التهشير بصورة عامة بالسرعة العالية في المعالجة الحاسوبية.

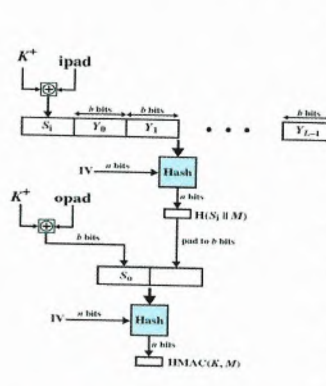


الشكل *(11)* مخطط لتابع تهشير عام

اعتمد أول تابع تهشير آمن *secure hash algorithm SHA-1* عام *1993* من معهد *NIST* الذي يتعامل مع كتلة مكونة من *512* بت وينتج قيمة تهشير *160* بت أو *20* بايت. وتم في عام *2002* اعتماد توابع تهشير بأطول قيمة تهشير مختلفة حسب القيمة العددية المرفقة مع الأسم: *SHA-256, SHA-384, SHA-512* بحث تحقق آمان أفضل. ومن أشهر توابع التهشير مختصر الرسالة نسخة 5 *MD5*.

خوارزمية *MAC based on Hash Function HMAC*

هي نوع من خوارزميات التحقق من الرسالة لكنها تميز عن خوارزمية *MAC* بسرعة المعالجة مثل توابع الهاش مع إضافة مفتاح سري للرسالة للتحقق من مصدر البيانات. في خوارزمية *HMAC* يُحسب الهاش من كتل الرسالة *Y* مضافاً لها المفتاح السري في أول كتلة في بدايتها S_i. ويتكرر حساب الهاش لكتلتين هما الهاش الناتج من المرحلة الأولى والمفتاح السري كما هو مبين في الشكل *(12)*.



الشكل *(12)* مخطط حساب *HMAC*

2 - 10 - التوقيع الرقمي Digital Signature

يتم تشكيل التوقيع الرقمي لنص كما يلي:

1. ضغط النص أوالرسالة باستخدام تابع تهشير *hash function* للحصول على مختصر النص *digest*.

2. تطبيق خوارزمية التشفير بالمفتاح العمومي على مختصر النص باستخدام المفتاح الخاص *private key* للموقّع للحصول على توقيعه الرقمي لهذا *1* النص.

3. لصق التوقيع الرقمي الناتج بالنص للحصول على نص موقع رقمياً.

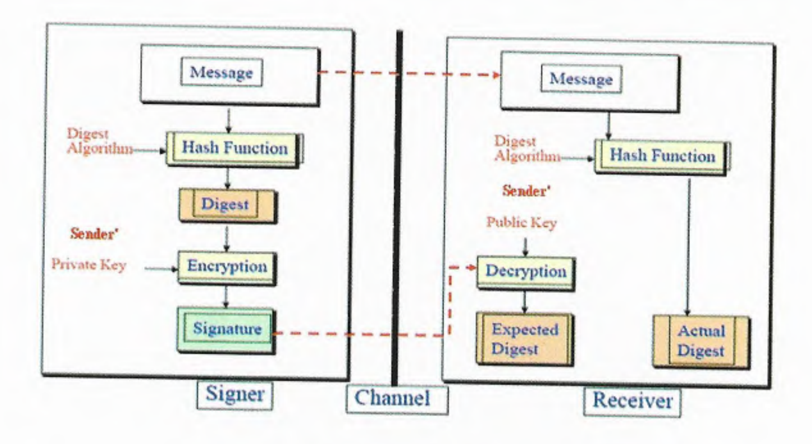
ويتم التحقق من التوقيع الرقمي على الشكل التالي:

1. فصل التوقيع الرقمي عن النص الموقع.

2. فك تشفير التوقيع الرقمي باستخدام خوارزمية التشفير بالمفتاح العمومي وباستخدام المفتاح العمومي *public key* للموقّع، وذلك للحصول على مختصر النص المتوقع *expected* التي قام الموقّع بحسابها عند التوقيع.

3. حساب مختصر النص من خلال تطبيق تابع التهشير ذاته الذي استخدمه الموقّع عند التوقيع.

4. المقارنة بين مختصر أو بصمة النص المحسوب في الخطوة *3* مع مختصر أو بصمة النص المفكوك في الخطوة *2*. وفي حال تساوي البصمتين يتم التحقق من سلامة بيانات النص *data integrity*، كما يدل ذلك على على صحة هوية الموقّع أيضاً كونه هو الوحيد الذي يمتلك المفتاح الخاص.

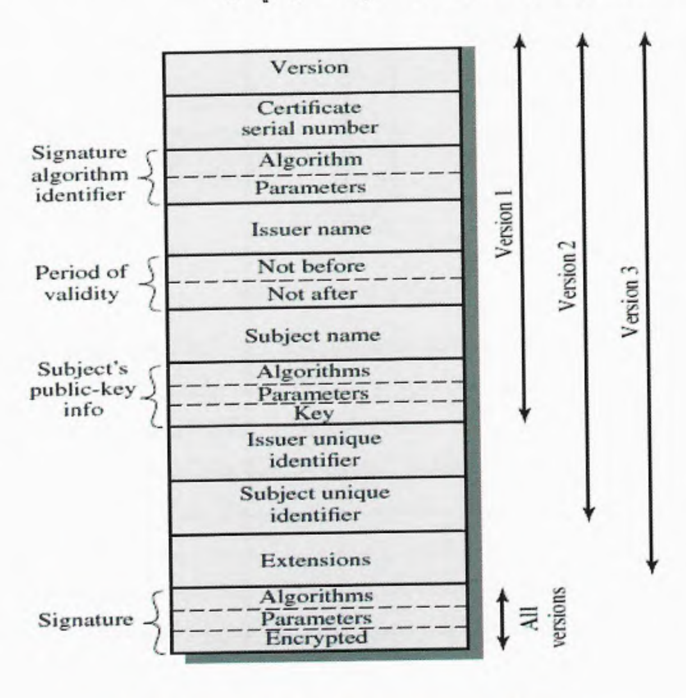


الشكل *(13)* التوقيع الرقمي

ملاحظة: بالإضافة إلى كون التوقيع الرقمي يحقق متطلبات التحقق من الرسائل *message authentication*، فإنه يقدم خدمة إضافية وهي عدم الإنكار *non-repudiation*، أي أن الموقّع لن يتمكن من إنكار توقيعه وكونه مصدراً للبيانات الموقعة رقمياً. لكن هنا لابد من الربط بين المفتاح العمومي وهوية صاحب المفتاح، ويتم ذلك من خلال شهادة يصدرها طرف ثالث يسمى سلطة إصدار الشهادة *(CA)* *certificate authority* مهمتها الأساسية الربط بين المفتاح العمومي لصاحب الشهادة وهويته. وتتضمن أيضاً التوقيع الالكتروني للجهة التي أنشأتها *CA*. ويطلق في القانون السوري اسم "مزود خدمات التوقيع الألكتروني" على سلطة إصدار الشهادة *CA*.

يبين الشكل *(14)* صيغة الشهادة الرقمة *digital certificate* وفق المعيار *X. 509* والتي تتضمن الإصدار *version* حيث يوجد ثلاثة إصدارات، والرقم التسلسلي للشهادة، والخوارزمية المستخدمة لإنشاء توقيع *CA* واسم سلطة إصدار الشهادة *CA* ومدة صلاحيتها واسم الكيان

أو الشخص الذي صدرت له الشهادة والمفتاح العمومي له وخوارزمية التشفير بالمفتاح العمومي، ومعرف فريد للمصدر وأخر للكيان او الشخص الذي صدرت له الشهادة وهما اختياريان. والتوقيع الرقمي لسلطة إصدار الشهادة والمفتاح العمومي لها.



الشكل *(14)* صيغة الشهادة الرقمية وفق المعيار *X. 509*

الفصل الثالث

التحقق من المستخدم *User Authentication*

التحقق من المستخدم هي العملية التي يتم فيها التحقق من الهوية التي يدعيها المستخدم سواء أكان شخصاً أو كياناً حاسوبياً. ولهذه العملية مرحلتان:

- **مرحلة التعريف** ***identification step***: يقوم فيها النظام الأمني بالتعرف على مُعرف المستخدم (مثلاً اسم المستخدم)
- **مرحلة التحقق** ***verification step***: توليد معلومات التحقق والتي تدعم الربط بين شخص المستخدم وأداة التعريف أو المعرف عنه (كلمة المرور *password*).

يختلف تحقق المستخدم عن تحقق الرسالة الذي يبين سلامة محتوى الرسالة وصحة مصدرها. ويعتبر تحقق المستخدم اللبنة الأساسية لأمن النظم المعلوماتية وخط الدفاع الأول فيها. وهو أيضاً الأساس لمعظم أنظمة التحكم بالدخول *access control*.

3 - 1 - وسائل التحقق من المستخدم

يتواجد أربع وسائل للتحقق من هوية المستخدم استناداً إلى:

- شيء يعرفه الفرد مثل كلمة المرور أو الرقم الشخصي *personal pin identification number*
- أو يمتلكه مثل بطاقة مغناطيسية او بطاقة ذكية أو مفتاح الكتروني

- ويمثله (هو) المزايا البيومترية الثابتة مثل بصمة الإصبع *fingerprint*، الشبكية *retina*، القزحية *iris* أو مميزات الوجه *face features*.
- أو يفعله المزايا البيومترية الديناميكية مثلك التعرف على المتكلم من صوته أو من خط الكتابة

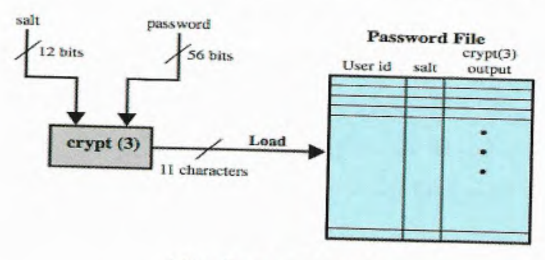
3 - 2 - التحقق المستند إلى كلمة المرور

يعتبر نظام كلمة المرور (أو كلمة السر) المستخدم على نطاق واسع هو خط الدفاع الأمامي ضد المتطفلين *intruders*. وتقريباً جميع الأنظمة متعددة المستخدمين تتطلب أن يوفر المستخدم اسم معرف *ID* وكلمة مرور.

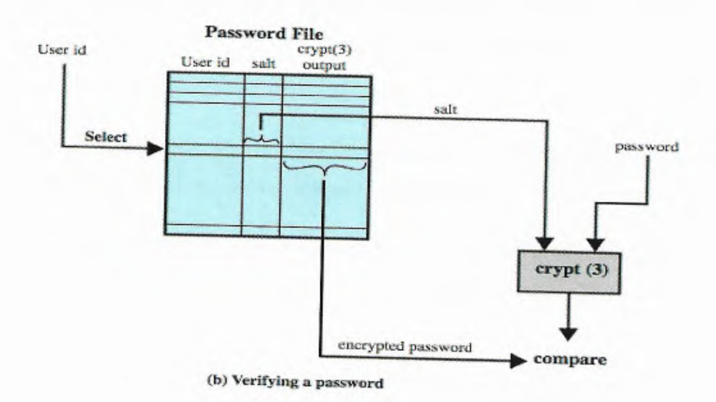
يقارن النظام كلمة المرور المدخلة مع كلمة المرور المخزنة مسبقاً لمعرف *ID* للمستخدم والتي يحتفظ بها في ملف كلمة مرور النظام *system password file*. في المقابل، يحدد المعرف *ID* الامتيازات *privileges* الممنوحة للمستخدم.

3 - 3 - تقنية كلمة المرور المهشرة

يبين الشكل *(15)* تقنية كلمة المرور المهشرة *hashed password technique*.



(a) Loading a new password



(b) Verifying a password

الشكل *(15)* تقنية كلمة المرور المهشرة

كانت تستخدم هذه الطريقة على نطاق واسع تقريباً في كل الخدمات التي تستخدم نظام تشغيل *unix* إضافة لنظم التشغيل الأخرى. وتستخدم على الشكل التالي في طور تعريف كلمة مرور جديدة لمستخدم ما:

- كلمة المرور المدخلة مكونة من 8 محارف (مرمزة بـ ASCII)
- هناك قيمة ثنائية مكونة من 12 بت والتي تسمى القيمة الملحية salt value تعطى لكل مستخدم
- كلمة المرور والقيمة الملحية يشكلان معاً دخل لخوارزمية تشفير تسمى crypt(3) وهي نسخة معدلة من DES. تمثل كلمة المرور مفتاح التشفير (56 بت) والقيمة الملحية تسخدم ضمن خوارزمية DES ودخل المشفر عبارة عن كتلة من 64 بت صفرية ويتم تشفيرها 25 مرة متتابعة.
- خرج المشفر النهائي 64 بت يُشكل منها 11 محرف تمثل كلمة المرور المهشرة والتي تخزن ضمن ملف يتضمن كلمة المرور المهشرة مع مُعرف المستخدم ID والقيمة الملحية المعطاة له.

في طور التحقق verification يتم مايلي:

- يقوم المستخدم بإدخال المعرف الخاص به ID وبدوره يقوم نظام التشغيل بانتقاء القيمة الملحية المقابلة لها
- القيمة الملحية وكلمة المرور المدخلة من المستخدم يشكلان معاً دخلاً لخوارزمية التشفير crypt(3) والتي تعطي كلمة المرور المهشرة.
- تتم المقارنة بين كلمة المرور المهشرة المحسوبة وكلمة المرور المهشرة المخزنة وفي حال التساوي تُقبل كلمة المرور ويسمح للمستخدم باستخدام النظام المعلوماتي.

لهذه الطريقة عدد من نقاط الضعف بحيث يمكن التكهن بكلمة المرور بسهولة نسبياً من خلال تجريب قاموس من كلمات السر لكافة قيم القيمة الملحة (4096 قيمة).

تم تحسين هذه الطريقة من خلال:

- استخدام *48* بت قيمة ملحية
- كلمة مرور غير محدودة الطول
- يكرر التشفير *1000* مرة
- كلمة المرور المهشرة الناتجة تساوي *128* بت.

خوارزمية *bcrypt* تستخدم خوارزميات تهشير تستعمل *128* بت قيمة ملحية وتعطي *192* بت كلمة مرور مهشرة.

يمكن تجنب حالات كسر *cracking* كلمات المرور إذا تم انتقاء كلمة مرور مكونة من ثمانية حروف منتقاة عشوائياً تتضمن حروف وأرقام وعلامات تنقيط *punctuation marks*. بعض النظم الحاسوبية تستخدم خوارزمية لفحص كلمة المرور المنتقاة (عند تعريف المستخدم) بحيث توازن بين قبول الكلمة وإمكانية تذكرها من المستخدم ومقاومتها للكسر.

3 - 4 - التحقق باستخدام البطاقات

هناك العديد من البطاقات البلاستيكية المغناطيسية التي تحتوى على شريط مغناطيسي أو البطاقات الذكية التي تتضمن شريحة إلكترونية (ذاكرة أو معالج وذاكرة) والتي تستخدم للدخول للنظم الحاسوبية مع او بدون رقم شخصي *personal identification number pin*.

3 - 5 - التحقق باستخدام المميزات البيومترية

بعض الأنظمة الحاسوبية تستخدم بصمة الإصبع أو بصمة قزحية العين بدلاً عن كلمة المرور. كما أن هناك نظم متطورة تستخدم بطاقات ذكية تتضمن البصمة مشفرة بتقنية التشفير بالمفتاح العمومي باستخدام مفتاح سري خاص للمستخدم (خلال طور التعريف) ويتم التحقق من المستخدم (في طور التحقق) بوضع البطاقة ضمن قارئ في النظام وقراءة البصمة الحية عبر قارئ بصمات الكتروني. تتم المقارنة بين مميزات البصمة الحية مع تلك المخزنة في البطاقة بعد قراءتها وفك تشفيرها بخوارزمية المفتاح العمومي باستخدام المفتاح العمومي للمستخدم المخزن على نفس البطاقة. ويمنح المستخدم حق الدخول في حال نجاح المقارنة بين البصمتين.

لازالت مسألة استخراج المميزات الحيوية أو البيومترية للبصمة أو القزحية أو الوجه موضع بحث حتى هذا الوقت بهدف استخراج المميزات الحيوية التي تحقق أقل نسبة خطأ في عملية التحقق.

3 - 6 - التحقق عند الدخول عن بعد

لتجنب التقاط كلمة المرور أو كلمة المرور المهشرة عبر الشبكات الحاسوبية عند الدخول لأنظمة حاسوبية عن بعد يتم التحقق من الدخول عن بعد *remote user authentication* باستخدام طريقة تسمى *challenge-response protocol* والتي تعمل وفق ما يلي:

- يرسل المستخدم كلمة مُعرفه ID إلى المخدم البعيد الذي يمتلك كلمة المرور المهشرة $h(p)$ لهذا المستخدم المخزنة في طور تعريف أو تسجيل المستخدم على النظام الحاسوبي.
- يستجيب المخدم بإرسال استجابة r وهي عبارة عن رقم عشوائي يسمى $challenge$ ويحدد تابعين $f()$ و $h()$ لاستخدامهما من طرف المستخدم.
- يقوم المستخدم بحساب التابع $f(r,h(p))$ الذي يتضمن القيمة العشوائية r وكلمة المرور المهشرة $h(p)$ للمستخدم.
- يقارن المخدم قيمة $f(r,h(p))$ المستلمة من المستخدم مع قيمة $f(r,h(p))$ المحسوبة لديه وفي حالي التساوي يتم السماح للمستخدم بالدخول للنظام الحاسوبي.

تقاوم هذه الطريقة الهجوم على المخدم البعيد للحصول على كلمة المرور كون المخدم يخزن كلمة المرور المهشرة $h(p)$ للمستخدم، كما تقاوم التقاط $h(p)$ على الشبكة (وإعادة ارسالها للمخدم) وذلك بقيام المستخدم بإرسال كلمة المرور المهشرة ضمن تابع f مع كلمة عشوائية r جديدة تُعطى من المخدم مع كل جلسة اتصال.



الفصل الرابع

مهددات النظم المعلوماتية *Threats to Information System*

أمن النظم المعلوماتية يؤدي أربع وظائف مهمة للمؤسسة:

- يحمي قدرة المؤسسة على العمل
- تمكين التشغيل الآمن للتطبيقات المركبة على النظم المعلوماتية
- يحمي البيانات التي تقوم المؤسسة بجمعها واستخدامها
- حماية مكونات المنظومة المعلوماتية المستخدمة في المؤسسة

المهددات *threats*: تشكل خطر دائم على الممتلكات

تقوم الإدارة، من خلال دراسة كل فئة من فئات المهددات، بحماية المعلومات بفعالية من خلال وضع السياسة الأمنية اللازمة وتعليم وتدريب الطاقم الفني في الشركة وباستخدام الإجراءات التقنية اللازمة للحماية.

4 - 1 - مهددات النظم المعلوماتية

1. الأخطاء البشرية - أخطاء العاملين

2. خروقات الملكية الفكرية

3. الاختراق المتعمد للنظم الحاسوبية من أشخاص غير مخولين بذلك

4. الابتزاز باستخدام معلومات مسروقة

5. التخريب المتعمد للمعلومات أو الانظمة الحاسوبية

6. سرقة الحواسيب عمداً

7. هجوم متعمد على البرمجيات

8. قوى الطبيعة *Forces of nature* (مثل البرق، الزلزال، الفيضان، الرياح، الغبار، التلوث،...)

9. انحراف في جودة الخدمات من مزودي الخدمات -(انترنت وكهرباء...)

10. أخطاء في العتاديات *hardware failures or errors*

11. أخطاء في البرمجيات *software failures or errors*

12. قدم التكنولوجيا المستخدمة في الانظمة ***technological obsolescence***

الهجوم *attack*: الفعل الذي يستغل الضعف في النظام المعلوماتي الخاضع للضوابط الأمنية.

الضعف *vulnerability*: هو ضعف يتم التعرف عليه في منظومة حيث لا يوجد ضوابط أمنية فيها أو أن الضوابط لم تعد فعالة (غير محدثة *not updated*).

التهديدات دائماً موجودة، لكن تحدث الهجمات نتيجة استغلال نقاط الضعف.

4 - 2 - الأخطار التي تتعرض لها النظم المعلوماتية

1. كسر كلمة المرور *password cracking*

2. هجوم الرجل في المنتصف *man in the middle attack*

3. الهندسة الاجتماعية *social engineering*

4. الأفراد أو الناس *people* (الموظفون العاملون على المنظومات المعلوماتية)

5. هجوم الحرمان من الخدمة *denial of service (DoS) attack*

6. الاختراق *intrusion*

7. البرمجيات الخبيثة *malicious code*

8. المتنصت *sniffer* (برنامج أو جهاز الذي يراقب البيانات المارة عبر الشبكة)

تشكل الأخطار الناجمة عن الموظفين (الحلقة الأضعف في مكونات المنظومة المعلوماتية) أو الافراد داخل المؤسسة *70%* من إجمالي الأخطار الممكن أن تتعرض لها النظم المعلوماتية. وتعتبر البرمجيات الجزء الأصعب في الحماية بينما البيانات هي الجزء الأكثر قيمة في المنظومة.

4 - 3 - البرمجيات الخبيثة

البرمجيات الخبيثة *malicious software* أو اختصاراً *malware* هي برمجيات ضارة تستغل نقاط الضعف في الحاسوب لتخترقه دون رضا مستخدم الحاسوب. فمثلاً الفيروس *virus* برنامج صغير، يُكتب بغرض إلحاق الضرر بحاسوب آخر، أو السيطرة عليه، بحيث يصيب الملفات التشغيلية الموجودة في الجهاز المستهدف، وعند تنفيذ هذه الملفات يقوم الفيروس بالعمل وتنفيذ المهام التي عمل من أجلها والتي غالباً ما تكون لأغراض تخريبية كحذف الملفات والعبث بنظام التشغيل والانتقال من حاسوب لآخر، وربما أيضاً يقوم بنسخ نفسه داخل نفس الجهاز، وتتم برمجة هذه الفيروسات بواسطة مبرمجين هواة أو محترفين لأسباب بهدف الحاق الضرر بأجهزة الحواسيب أوتكون لتحقيق مكاسب مالية. ولايقتصر دور الفيروسات على ملفات تدمير النظام بل ربما يتعداها لتدمير البطاقة الرئيسية *motherboard* في الحاسوب من خلال التلاعب بالإعدادات في برنامج *BIOS* الذي يتحكم بالحاسوب بالكامل. لذلك تأثير الفيروسات علي جهاز الكمبيوتر يختلف من فيروس لآخر.

تصنف البرمجيات الخبيثة ضمن ثلاثة مجموعات:

- مجموعة البرامج البسيطة *virus, worm, logic bomb*
- مجموعة برامج التجسس: *trojan horse, backdoors, spywares*
- المجموعة التي تولد الفيروسات وهي الأخطر كونها تسيطر على مجموعة من الحواسيب مثل *rootkit, flooder, auto-rooter*.

مؤشرات الإصابة بالفيروسات

يمكن ملاحظة أشياء غريبة تحدث في الجهاز عند إصابته بفيروس مثل:

- تظهر رسائل خطأ غير معروفة ومربعات حوار غير مألوفة
- بعض البرامج لا تعمل بشكل صحيح وأخرى لا تعمل على الاطلاق.
- بطئ شديد في عمل الجهاز وتباطئ في تنفيذ الأوامر
- امتلاء القرص الصلب بما لا يتناسب مع عدد وحجم الملفات الموجودة عليه
- إضاءة لمبة سواقة القرص الصلب أو سواقة CD، دون أن تقوم بعملية فتح أو حفظ ملف
- عند تصفح شبكة الإنترنت يلاحظ أن الصفحات التي تختارها لا تعمل بشكل صحيح، وقد يتم التحويل إلى صفحات أخرى، وأيضاً يلاحظ أن الجهاز يقوم بارسال رسائل لقائمة بريدية بدون علمك، بشكـــل عام.

الحماية من البرمجيات الخبيثة

- لابد من وجود برنامج حماية من الفيروسات في جهازك مع تحديثه بشكل دوري وإلا لافائدة من وجوده.
- عدم فتح مرفقات البريد الإلكتروني الواردة من مصادر غير معروفة و تلك التي تنتهي بـ exe أو bat أو أي امتداد لا تعرفه.

- قبل نسخ أي ملف من واسطة تخزين خارجية يجب فحصها ببرنامج الحماية.
- عدم الدخول لمواقع غير موثوقة على الانترنت
- تفعيل الجدار الناري (إغلاق البوبات غير المرغوب بتشغيلها) في نظام التشغيل وفي برامج الحماية.

الفصل الخامس

أمن الشبكات *Network Security*

5 - 1 - نظام كشف الاختراق *IDS*

يعرف المعيار *RFC2828* كشف الاختراق *intrusion detection*بالشكل التالي: خدمة أمن تراقب وتحلل أحداث النظام بهدف اكتشاف محاولات الوصول إلى موارد النظام بأسلوب غير مشروع، وإصدار تنبيه في الزمن الحقيقي أو شبه الحقيقي في حال اكتشاف إحدى هذه المحاولات.

يمكن تصنيف أنظمة كشف الاختراق *intrusion detection system* كما يلي:

1. نظام كشف الاختراق الخاص بحاسوب *Host-based IDS (HIDS):* يراقب خصائص حاسوب معين بمفرده أثناء عمله لاكتشاف النشاطات المشبوهة عليه.

2. نظام كشف الاختراق الخاص بشبكة *Network-based IDS (NIDS):* يراقب البيانات أثناء مرورها في الشبكة ويحلل محتوى الطرود الخاصة ببروتوكولات طبقات الشبكة والنقل والتطبيقات بهدف اكتشاف النشاطات المشبوهة.

3. نظام كشف الاختراق الموزع أو الهجين *distributed or hybrid (IDS):* يجمّع المعلومات الواردة من مجموعة من الحساسات في محلل مركزي، وتكون الحساسات متنوعة بين حساسات خاصة بحواسيب معينة *host-based* وحساسات خاصة بالشبكة *network-based*، بحيث يتم تحديد نشاطات الاختراق والاستجابة لها بشكل أفضل من النوعين السابقين.

يتألف نظام كشف الاختراق من ثلاثة مكونات وفق منطق عمله، وهي:

1. الحساسات *sensors* التي تجمع البيانات.

2. المحللات *analyzers* التي تحدد فيما إذا حدث اختراق.

3. واجهة المستخدم *user interface* التي تسمح بالإطلاع على خرج النظام أو التحكم بطريقة عمله.

يمكن تصنيف المقاربات التحليلية *analysis approaches* في انظمة كشف الاختراق كما يلي:

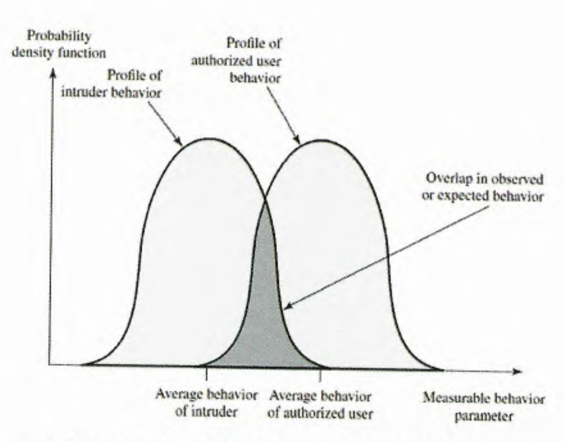
1. كشف الشذوذ *anomaly detection*:

- يتم جمع البيانات التي ترتبط بسلوك المستخدمين الشرعيين *legitimate users* خلال مدة معينة.
- يتم تحليل سلوك المستخدمين الحاليين لتحديد فيما إذا كان يصنف سلوكاً لمستخدم شرعي وإلا فهو سلوك لمخترق.
- تتنوع طرقه بين إحصائية ومبنية على المعرفة ومبنية على تعلم الآلة.

2. الكشف الاستدلالي أو المبني على التوقيع *signature / heuristic detection*:

- يستخدم مجموعة من نماذج البيانات الخبيثة *malicious data patterns* أو قواعد الهجوم *attack rules* المعروفة ويقارنها مع السلوك الحالي.
- معروف أيضاً باسم الكشف عن سوء الاستخدام *misuse detection*.
- يمكنه فقط تحديد الهجمات المعروفة والتي لديه نماذج أو قواعد ترتبط بها.

يحتاج مديرو نظم كشف الاختراق لتحقيق التوازن في إعداد قواعد الكشف وإلا أعطى نظام كشف الاختراق تنبيهات خاطئة *false alarms*. عندما يتجه مدير النظام إلى وضع قواعد صارمة تضع الكثير من القيود على سلوك المستخدم الشرعي، من الممكن أن يصدر نظام كشف الاختراق تنبيهات موجبة خاطئة *false positives* حيث يدل التنبيه الموجب على وجود اختراق وقد يكون المستخدم شرعياً ويقوم بنشاطات مسموحة. من ناحية أخرى، عندما تكون قواعد الكشف متهاونة قد يصدر نظام كشف الاختراق تنبيهات سالبة خاطئة *false negatives* بمعنى وجود اختراق لم يتمكن من كشفه. يبين **خطأ! لم يتم العثور على مصدر المرجع.** التداخل المحتمل بين سلوك المخترق والمستخدم الشرعي بحيث قد تؤدي القواعد المعرفة لنظام كشف الاختراق عند تقارب السلوكين إلى إصدار التنبيهات الخاطئة *false alarms*.



الشكل *(16)* التداخل بين سلوكيات المخترق والمستخدم الشرعي

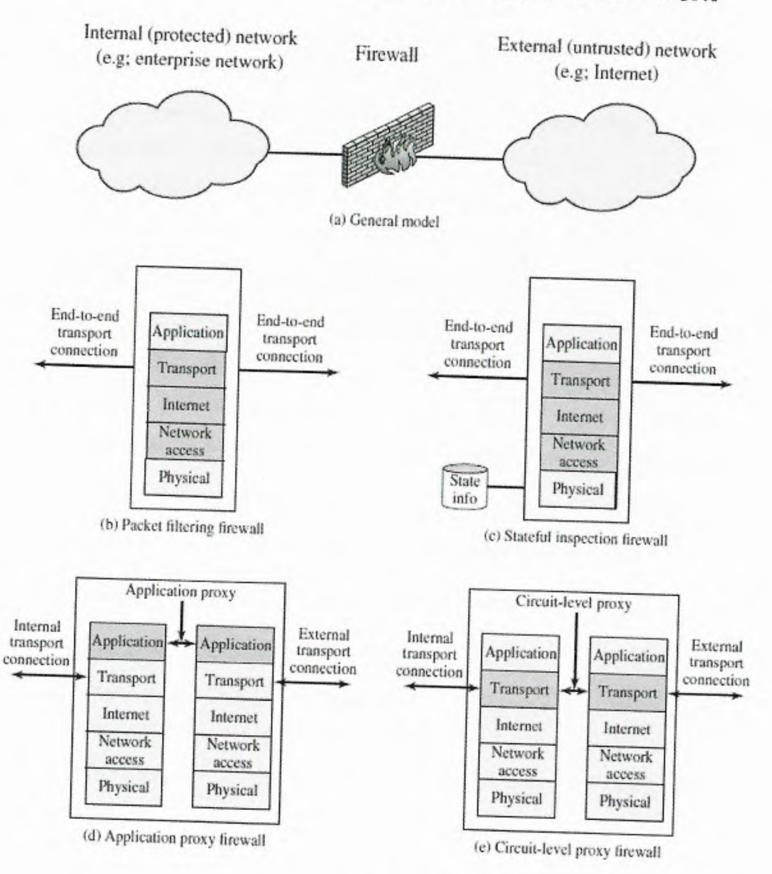
5 - 2 - الجدار الناري

الجدران النارية *firewalls* أدوات فعالة لحماية الشبكات المحلية *LAN* عند وصلها بالإنترنت أو بشبكات خارجية. حيث يتم تشغيلها عند النقاط الفاصلة بين الشبكة المحلية والعالم الخارجي لإنشاء وصلات مضبوطة *controlled links* تعزل الأنظمة الداخلية عن الشبكات الخارجية.

يمكن تلخيص الأهداف التصميمية للجدار الناري كما يلي:

- يجب أن تمر كل البيانات التي يتم تبادلها بين الداخل والخارج وفي الجهتين عبر الجدار الناري.
- يتم السماح بمرور البيانات المشروعة فقط وفق قواعد سياسة الأمن الخاصة بالشبكة المحلية.
- يجب أن يكون الجدار الناري ذاته منيعاً ضد الاختراقات.

يبين الشكل *(17)* أنواع الجدران النارية:



الشكل *(17)*: أنواع الجدران النارية

الجدار الناري المرشح للطرود *Packet Filtering Firewall*

يطبق هذا الجدار الناري قواعده على كل طرود بروتوكول الإنترنت *IP Packets* الداخلة إلى الشبكة المحلية والخارجة منها. تعتمد هذه القواعد بشكل عام على المطابقة مع بيانات موجودة في ترويسات طرود بروتوكول الإنترنت *IP* أو بروتوكول طبقة النقل *TCP*، مثل عنوان بروتوكول الإنترنت *IP Address* الخاص بالمصدر أو عنوان طبقة النقل (رقم البوابة) الخاص بالوجهة على سبيل المثال. إما أن تؤدي هذه المطابقة إلى السماح بتمرير الطرد إلى وجهته، أو قد تؤدي إلى إيقافه وحذفه من قبل الجدار الناري. هناك سياستان لعمل هذا النوع من الجدران النارية:

1. سياسة الحذف *Discard* التي تمنع كل الطرود من المرور كقاعدة أساسية ثم تسمح بمرور طرود تحقق استثناءات وفق قواعد إضافية.
2. سياسة التمرير *Forward* التي تسمح لكل الطرود بالمرور كقاعدة أساسية ثم تمنع طرود معينة من المرور وفق قواعد إضافية.

الجدار الناري المتحقق من الحالة *Stateful Inspection Firewall*

هو جدار ناري مرشح للطرود *packet filtering firewall* تم إغناؤه بإمكانية إنشاء مجلد لتخزين بيانات حول اتصالات البروتوكول *TCP* المتجهة إلى خارج الشبكة بكاملها وليس فقط طروداً إفرادية. وبالتالي يؤمن هذا النوع منع الهجمات التي تنفذ على اتصال *TCP* بكامله.

بوابة مستوى التطبيقات *Application-Level Gateway*

يسمى هذا الجدار الناري أيضاً بوسيط التطبيقات *Application Proxy*. تعمل بوابة مستوى التطبيقات كنقطة مرحلية *relay* بالنسبة لحركة مرور البيانات على مستوى التطبيقات، حيث يتصل بها

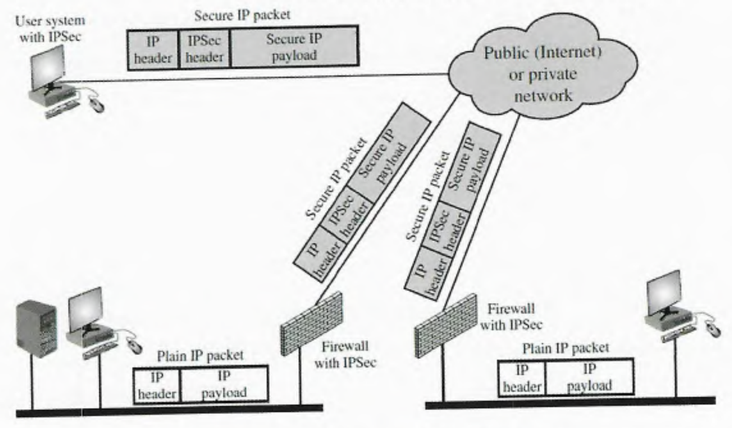
المستخدم بهدف استخدام تطبيق خارج الشبكة المحلية، لتقوم بدايةً بالتحقق من هويته، ثم في حال سمحت قواعدها تقوم بلعب دور الوسيط في تمرير مقتطعات بروتوكول النقل *TCP Segments* بين المستخدم والتطبيق البعيد، ويكون لديها القدرة بالتالي على مراقبة البيانات المتبادلة بين الطرفين وتطبيق قواعد الترشيح عليها. يعتبر هذا النوع من الجدران النارية الأكثر أمناً، ولكنه قد يسبب بطءاً في التطبيقات الشبكية بسبب المعالجة الإضافية للبيانات التي يتطلبها في كل اتصال مع تطبيق خارج الشبكة المحلية.

بوابة مستوى الدارات *Circuit-Level Gateway*

يسمى هذا الجدار الناري أيضاً بوسيط مستوى الدارات *Circuit-Level Proxy*. تقوم بوابة مستوى الدارات من أجل اتصال معين مع تطبيق خارج الشبكة المحلية بإنشاء اتصالي *TCP* أحدهما مع الحاسب الداخلي الذي طلب الاتصال والآخر مع الحاسب الخارجي الذي يقدم الخدمة المطلوبة. لا يقوم وسيط مستوى الدارات بفحص محتويات مقتطعات بروتوكول النقل *TCP* وإنما يقرر فقط فيما إذا كان سيسمح بمرورها بين الطرفين وفق قائمة لديه تحدد الاتصالات المسموحة. يستخدم عادةً هذا النوع من الجدران النارية عندما يكون مستخدمو الشبكة المحلية الداخليون موثوقين.

يمكن أن يكون الجدار الناري مخصصاً لحماية حاسوب وحيد *Host-Based Firewall* بدلاً من شبكة بكاملها. يتوفر هذا النوع من الجدران النارية في نظم التشغيل أو على شكل حزمة برمجية مستقلة. تكون مهمته في معظم الحالات ترشيح الطرود الواردة إلى الحاسب من الشبكة الموصول عليها أو من الإنترنت، ويتم تشغيله عادةً على المخدمات.

يمكن الاستفادة من الجدار الناري كجهاز شبكي يصل الشبكة المحلية بالعالم الخارجي (بشبكة الإنترنت عادةً) لتحقيق خدمات أخرى بالإضافة إلى مهمته الأساسية وهي ترشيح حركة مرور البيانات من وإلى الشبكة. من أهم هذه الخدمات الإضافية إنشاء شبكة خاصة افتراضية *VPN (Virtual Private Network)* بين شبكتين محليتين أو بين مستخدم بعيد وشبكة محلية، وذلك فوق الإنترنت أو أي شبكة واسعة المساحة *WAN (Wide Area Network)*. يقوم الجدار الناري في هذه الحالة بتطبيق بروتوكول *IPSec* على طرود *IP* بحيث تضاف إليها خدمات الأمن، ولاسيما تشفير البيانات وكود التحقق من سلامة الطرد ومصدره، عند خروجها من الشبكة المحلية، ويعيدها إلى صيغتها الأساسية كطرود *IP* بدون خدمات أمن عند دخولها إلى الشبكة المحلية. سنشرح البروتوكول *IPSec* كأحد بروتوكولات أمن الإنترنت في الفصل القادم. يبين الشكل *(18)* أحد سيناريوهات إنشاء شبكة خاصة افتراضية ودور الجدار الناري في هذا السيناريو كجهاز مسؤول عن خدمات البروتوكول *IPSec*.



الشكل *(18)*: سيناريو شبكة خاصة افتراضية *(VPN)*

هناك عدة طرق لإضافة الجدار الناري إلى منظومة معلوماتية سواء لحماية حاسوب وحيد أو لحماية شبكة بكاملها. تختلف هذه الطرق من حيث البنية الشبكية وموقع إضافة الجدار الناري ونمطه. فيما يلي قائمة تلخص البنى المعلوماتية المختلفة التي تعتمد على الحماية بجدران النار:

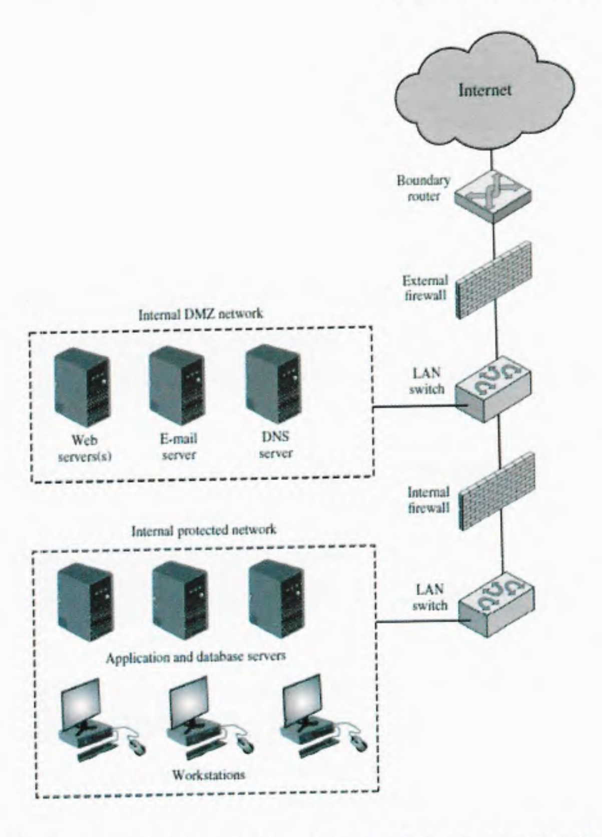
1. **الجدار الناري العامل على حاسوب *host-resident firewall*:** يشمل هذا الصنف برمجيات الجدار الناري الشخصي *personal firewall*، وبرمجيات الجدران النارية المخصصة للمخدمات. يمكن استخدام الجدار الناري من هذا الصنف وحده أو كجزء من بنية حماية مبنية على عدة جدران نارية.

2. **موجه الترشيح *screening router*:** موجه وحيد بين الشبكة الداخلية والشبكة الخارجية، ويعمل كجدار الناري مرشح للطرود. تناسب هذه البنية تطبيقات الشبكات المكتبية الصغيرة والمنزلية *small office / home office (SOHO)*.

3. **نقطة دفاع وحيدة داخلية *single bastion inline*:** جهاز جدار ناري وحيد بين موجه خارجي وموجه داخلي. من الممكن أن يكون نوع هذا الجدار الناري مرشح متحقق من الحالة و / أو بوابة على مستوى التطبيقات. يعتبر هذا الجدار الناري الحل النمطي لمؤسسة صغيرة أو متوسطة الحجم.

4. **نقطة دفاع وحيدة متشعبة *single bastion T*:** مشابه لنقطة الدفاع الوحيدة الداخلية ولكن له واجهة شبكية ثالثة توصل إلى شبكة فرعية من الشبكة الداخلية تسمى المنطقة منزوعة السلاح *demilitarized zone (DMZ)* حيث توضع المخدمات المسموح الوصول إليها من الشبكة الخارجية. يعتبر هذا الجدار الناري الحل النمطي لمؤسسة متوسطة أو كبيرة الحجم.

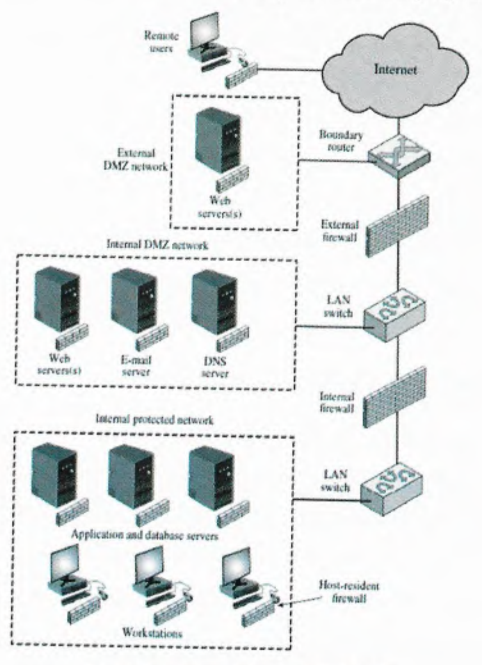
5. نقطة دفاع مضاعفة داخلية *double bastion inline*: يوضح الشكل *(19)* هذه البنية، حيث تنحصر المنطقة منزوعة السلاح بين جدارين ناريين. تعتبر هذه البنية شائعة في الشركات الكبيرة والمؤسسات الحكومية.



الشكل *(19)* بنية استخدام الجدران النارية من نوع نقطة الدفاع المضاعفة الداخلية

6. **نقطة دفاع مضاعفة متشعبة** ***double bastion T***: تبنى المنطقة منزوعة السلاح كشبكة مستقلة عن الشبكة الداخلية وتتصل على إحدى الواجهات الشبكية في نقطة الدفاع المثلة بالجدار الناري الأول بعد الموجه داخل شبكة المؤسسة. تعتبر هذه البنية أيضاً شائعة في الشركات الكبيرة والمؤسسات الحكومية، وقد تكون حتمية في بعضها.

7. **بنية الجدران النارية الموزعة** ***distributed firewall configuration***: يوضح الشكل *(20)* هذه البنية. تستخدم من قبل بعض الشركات الضخمة والمؤسسات الحكومية.



الشكل *(20)* بنية الجدران النارية الموزعة

5 - 3 - نظام منع الاختراق *IPS*

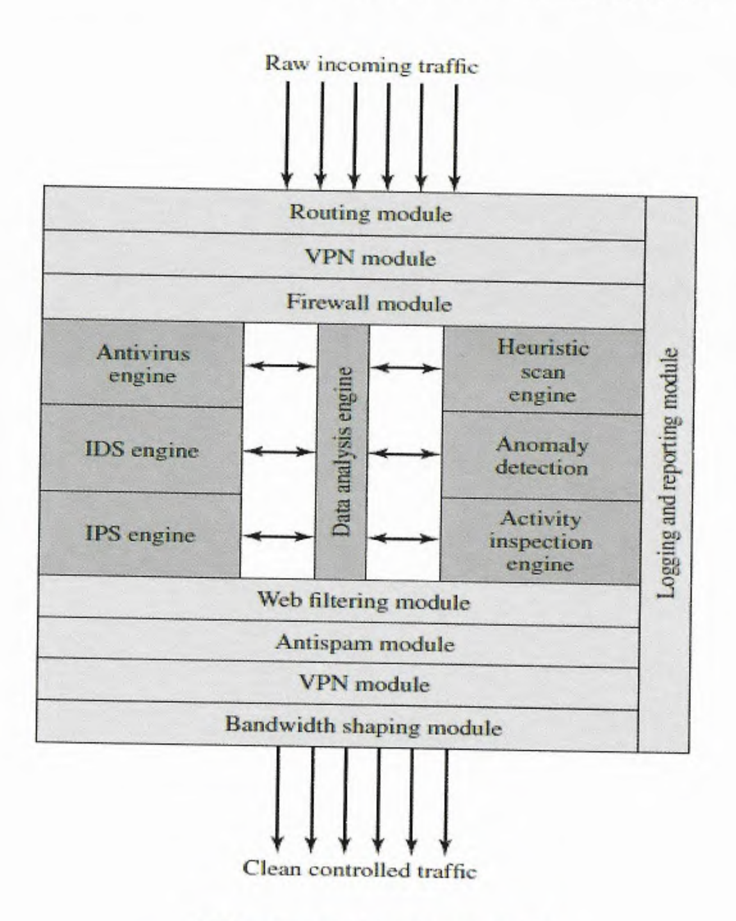
يسمى نظام منع الاختراق *intrusion prevention system* أيضاً بنظام كشف ومنع الاختراق *IDPS (intrusion detection and prevention system)*. نظام منع الاختراق هو نظام كشف اختراق معزز بإمكانية محاولة تعطيل أو منع النشاط الخبيث المكتشف.

يتبع نظام منع الاختراق التصنيفات ذاتها التي شرحناها مسبقاً عن نظام كشف الاختراق. يمكن أن يكون مخصصاً لحاسوب بمفرده *host-based* أو لشبكة كاملة *network-based* أو أن يأخذ الصيغة الموزعة أو الهجينة *distributed / hybrid*. من ناحية أخرى، من الممكن أن يستخدم طريقة كشف الشذوذ *anomaly detection* لتحديد السلوك الذي لا يمثل مستخدمين شرعيّين، أو طريقة الكشف الاستدلالي أو المبني على التوقيع *signature / heuristic detection* لتحديد السلوك الخبيث المعرف مسبقاً. أما القيمة المضافة التي يقدمها نظام منع الاختراق فهي قدرته على إيقاف حركة مرور البيانات كما يفعل الجدار الناري، إلا أنه يستفيد من إمكانية تنفيذ الخوارزميات المختلفة المطورة لأجل نظم كشف الاختراق ليحدد الوقت المناسب لإيقاف حركة مرور البيانات.

نظم إدارة التهديد الموحد *UTM (Unified Threat Management)*

تحصل مؤسسة ما على القدرة على الدفاع المتكامل عن أمن نظم معلوماتها في حال تمكنت من تنفيذ مجموعة متكاملة من نظم الحماية ضد البرمجيات الخبيثة *malicious software* والهجمات المبنية على الشبكات *network-based attacks* وذلك بتشغيلها على مستويات متتابعة ما تحتاجه من المرشحات وآليات الدفاع. يبين الشكل *(21)* تصميماً لجهاز يعمل كنظام يواجه مجموعة كبيرة من التهديدات عن طريق تكامل نظم الحماية التي سبق

وشرحناها في منظومة واحدة تدعى جهاز إدارة التهديد الموحد
.*UTM (Unified Threat Module) Appliance*



الشكل *(21)*: نظام إدارة التهديد الموحد *UTM*

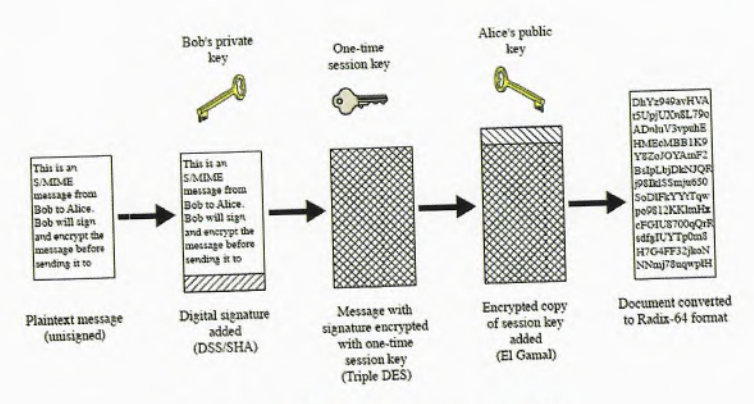
الفصل السادس

بروتوكولات أمن الإنترنت *Internet Security Protocols*

6 - 1 - أمن البريد الإلكتروني باستخدام بروتوكول *S / MIME*

يقدم بروتوكول التمديدات متعددة الأغراض لبريد الإنترنت *MIME: Multipurpose Internet Mail Extensions* حلاً لمحدودية بروتوكول نقل البريد البسيط *SMTP: Simple Mail Transfer Protocol* الذي لا ينقل سوى بيانات نصية من النمط *ASCII*، بينما يسمح استخدام البروتوكول *MIME* بنقل بيانات نصية وغير نصية وبأنماط مختلفة في محتوى رسالة البريد الإلكتروني. يسمح تصميم ترويسة الطرد *packet header* في البروتوكول *MIME* بإضافة أجزاء مخصصة لتقديم خدمات إضافية. نتج البروتوكول *S / MIME* عن استخدام هذه الميزة لإنتاج طرود آمنة *secure packets* يكون فيها محتوى الرسالة موقعاً رقمياً ومشفراً ومرمزاً مع ما تحتاجه خدمات الأمن هذه من إضافة أجزاء لإدارتها في ترويسة الطرد.

يهدف التوقيع الرقمي المبني على التعمية اللاامتناظرة *asymmetric encryption* إلى تقديم خدمة التحقق من الرسالة *message authentication* إلى المرسل إليه ليضمن سلامة محتوى الرسالة وصحة هوية مرسلها، كما يهدف إلى منع المرسل من إنكار رسالته *non-repudiation*. تتم التعمية باستخدام تقنية الظرف الرقمي *digital envelope* التي تسمح باستخدام مفتاح سري عشوائي مختلف مع كل رسالة إن تطلب الأمر. أما الترميز *encoding* فيهدف إلى تحويل الرسالة الموقعة والمعماة إلى بيانات نصية تتبع المعيار *ASCII* ليتم إرسالها بشكل صحيح باستخدام البروتوكول *SMTP*.



الشكل *(22)*: طريقة عمل *S / MIME* لدى المرسل

يمكن تنفيذ خدمات الأمن في البروتوكول *S / MIME* بشكل جزئي أو كلي وفق متطلبات تطبيق البريد الإلكتروني ومستخدميه. فيما يلي أنماط استخدام *S / MIME*:

1. بهدف تحقيق سرية الرسالة *message confidentiality* فقط. يمكن الاكتفاء بإنشاء الظرف الرقمي *digital envelope* ومن ثم الترميز *encoding*. لنحصل على ما يسمى البيانات المغلفة *enveloped data*.

2. بهدف تقديم إمكانية التحقق من الرسالة *message authentication* فقط للمرسل إليه. يمكن الاكتفاء بحساب التوقيع الرقمي *digital signature* ولصقه بالرسالة ومن ثم ترميز الرسالة وتوقيعها معاً. لنحصل على ما يسمى البيانات الموقعة *signed data*.

3. بهدف تقديم إمكانية التحقق من الرسالة *message authentication* فقط للمرسل إليه، مع فرضية أن المرسل إليه قد لا يمتلك إمكانية استخدام البروتوكول *S / MIME*، يمكن الاكتفاء بحساب التوقيع

الرقمي *digital signature* ولصقه بالرسالة ومن ثم ترميز التوقيع فقط بحيث يتمكن المرسل إليه من قراءة الرسالة على الأقل، لنحصل على ما يسمى البيانات الواضحة الموقعة *clear signed data*.

4. بهدف الاستفادة من كافة الخدمات الأمنية التي يقدمها البروتوكول *S / MIME* يتم حساب التوقيع الرقمي *digital signature* ولصقه بالرسالة، ثم إنشاء الظرف الرقمي *digital envelope* لكتلة الرسالة والتوقيع معاً، ثم تطبيق ترميز *encoding* على الظرف الرقمي الناتج ليصيغ رسالة نصية يقبلها البروتوكول *SMTP*.

في حال لم يمتلك المرسل إليه إمكانية استخدام البروتوكول *S / MIME* فلن يتمكن من الاستفادة إلا من الرسالة الواصلة بصيغة البيانات الواضحة الموقعة *clear signed data*، حيث يقوم بقراءة الرسالة الواضحة فقط دون معالجة التوقيع الرقمي المرفق. أما في حال كان لدى المرسل إليه إمكانية استخدام البروتوكول *S / MIME* فيقوم بالخطوات التالية كلياً أو جزئياً حسب نمط الحماية المطبق على الرسالة:

1. فك الترميز *decoding* ليحصل على الظرف الرقمي في حال وصول بيانات مغلفة *enveloped data*، أو ليحصل على الرسالة مع توقيعها الرقمي في حال وصول بيانات موقعة *signed data*، أو ليحصل على التوقيع الرقمي في حال وصول بيانات واضحة موقعة *clear signed data*.

2. فصل الرسالة المعماة عن المفتاح السري المعمى في حال استقبال ظرف رقمي، ثم فك التعمية اللاامتناظرة المطبقة على المفتاح السري باستخدام المفتاح الخاص *private key* للمرسل إليه، ثم فك التعمية المتناظرة المطبقة على الرسالة باستخدام المفتاح السري.

3. فصل الرسالة الواضحة عن التوقيع الرقمي، ثم حساب قيمة التهشير *hash value* للرسالة الواضحة، ثم فك التعمية اللاامتناظرة المطبقة في التوقيع الرقمي باستخدام المفتاح العمومي *public key* للمرسل للحصول على قيمة التهشير المرفقة مع الرسالة، ثم مقارنة قيمة التهشير المحسوبة لدى المرسل إليه مع قيمة التهشير المرفقة مع الرسالة. وفي حال التطابق يتحقق المرسل إليه من سلامة بيانات الرسالة وصحة هوية المرسل.

ملاحظة: يحتاج مستخدم البروتوكول *S / MIME* إلى زوج مفاتيح لامتناظرة بحيث يخزن مفتاحه الخاص *private key* لديه بشكل آمن، وينشر شهادة رقمية *digital certificate* ليتمكن المستخدمون الآخرون من استخدام مفتاحه العام *public key*.

6 - 2 - طبقة المقابس الآمنة SSL: Secure Sockets Layer

توفر طبقة المقابس الآمنة *SSL* حماية للبيانات خلال جلسة اتصال على مستوى طبقة النقل *transport layer*. قدمت هذه التقانة الأساس لمعيار أمن طبقة النقل *TLS: Transport Layer Security*. تعمل البروتوكولات الخمسة التالية في طبقة المقابس الآمنة *SSL*:

1. بروتوكول المصافحة *SSL Handshake Protocol* الذي يعمل على مستوى التطبيق *application layer* ويهدف إلى إنشاء الجلسة الآمنة.

2. بروتوكول تغيير محددات التعمية *SSL Change Cipher Spec Protocol* الذي يعمل على مستوى التطبيق *application layer* ويهدف إلى اعتماد محددات التعمية المتفق عليها مسبقاً في بروتوكول المصافحة.

3. بروتوكول الإنذار *SSL Alert Protocol* الذي يعمل على مستوى التطبيق *application layer* ويهدف إلى تبادل الإنذارات بين الخدم *server* والزبون *client* على طرفي جلسة *SSL*، وذلك في حال حدوث خطأ يتطلب إنهاء الجلسة أو مخالفة تتطلب التنبيه.

4. بروتوكول الخفقان *SSL Heartbeat Protocol* الذي يعمل على مستوى التطبيق *application layer* ويهدف إلى تحقق أحد طرفي الجلسة من كون الطرف الآخر لا يزال موجوداً ولم يتوقف عن العمل، كما يهدف من خلال رسائله إلى إبقاء الجلسة فعالة.

5. بروتوكول السجلات *SSL Record Protocol* الذي يعمل بين طبقتي النقل *transport layer* والتطبيق *application layer* ويهدف إلى توفير خدمات أمن المعلومات من خلال تعمية طرود التطبيق *application packets* بهدف إخفاء محتوياتها، وإضافة كود تحقق من الرسائل *MAC: Message Authentication Code* إلى الطرد ليسمح لمستقبله بالتحقق من محتواه ومن هوية مصدره. تستخدم كل بروتوكولات *SSL* الأخرى المذكورة أعلاه بروتوكول السجلات *SSL Record Protocol* لتحقيق خدمات أمن المعلومات كونها بروتوكولات تعمل على مستوى التطبيق *application layer*.

Handshake Protocol	Change Cipher Spec Protocol	Alert Protocol	HTTP	Heartbeat Protocol
Record Protocol				
TCP				
IP				

الشكل *(23)*: بروتوكولات طبقة المقابس الآمنة

من المفروض أن يتم تنفيذ *SSL* على شكل طبقة مستقلة في مكدس البروتوكولات *protocol stack* الخاص ببيئة *TCP / IP*، بحيث تعمل هذه الطبقة المستقلة فوق مستوى النقل *transport layer* وتحت مستوى التطبيقات *application layer* وبمعزل عن أي تطبيق. إلا أن الشكل المنتشر لتنفيذ تقانة *SSL* هو تنفيذ بروتوكولاتها المذكورة أعلاه في حزمة برمجية *software package* كإضافة *add-on* على تطبيق معين يتم استخدامها عند تحديد بوابة *port* معينة على مستوى النقل *transport layer* حيث يجب أن يكون البروتوكول الناقل لبيانات الجلسة هو بروتوكول التحكم بالنقل *TCP: Transmission Control Protocol* بالتحديد. على سبيل المثال، يتم تنفيذ تقانة *SSL / TLS* كإضافة إلى متصفح الويب *Web browser* بحيث يستخدمها عند تحديد البوابة رقم *443* على مستوى النقل، وهو ما ينتج عن عمل المستخدم عندما يبدأ عنوان الصفحة المطلوبة بالمقطع *https*، ويعبر هذا المقطع عن طلب حماية بيانات صفحة الويب أثناء النقل باستخدام *SSL / TLS*. يهدف فرض استخدام البروتوكول *TCP* حصراً مع طبقة *SSL* إلى إضافة خاصة الوثوقية *reliability* إلى خواص الأمن التي تؤمنها طبقة *SSL*، وهي السرية *confidentiality* وسلامة البيانات *data integrity* والتحقق من هوية مصدر البيانات *data source authentication*.

بروتوكول المصافحة *SSL Handshake Protocol*

يهدف هذا البروتوكول إلى إنشاء جلسات آمنة *secure sessions* على مستوى النقل *transport layer* للتطبيق الذي يستخدم *SSL* لحماية بياناته أثناء انتقالها على الشبكة. يعمل بروتوكول المصافحة وفق المنهجية التالية:

1. بناءً على طلب من الزبون *client* للاتصال الآمن باستخدام *SSL / TLS*، كأن يبدأ مستخدم الويب عنوان الصفحة المطلوبة بالمقطع

https على سبيل المثال، يرسل المخدم *server* شهادته الرقمية *digital certificate* إلى الزبون ليؤكد له أنه وصل إلى المخدم الصحيح، وبالتالي يجب أن يمتلك برنامج الزبون المفتاح العمومي الذي يساعد على التحقق من صحة الشهادة الرقمية للمخدم.

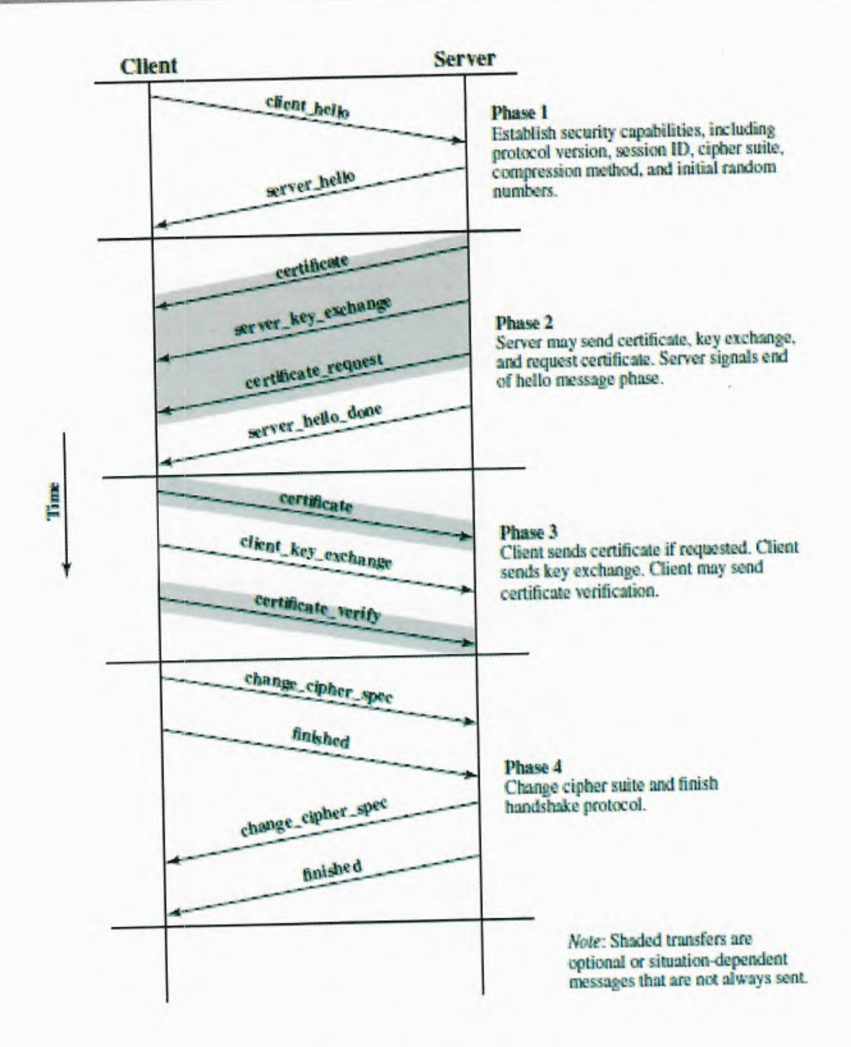
2. قد يطلب المخدم من الزبون شهادة رقمية تخص الزبون في تطبيقات معينة أو للاستجابة لطلبات معينة، كأن يفرض موقع مصرف ما على مستخدمه امتلاك شهادة رقمية في حال طلب خدمة يعتبرها المصرف حساسة مثل طلب فتح حساب ادخار على سبيل المثال.

3. تتم عملية تبادل لمفتاح سري مشترك بين المخدم والزبون، سيستخدمه المخدم لاحقاً في تعمية البيانات التي يرسلها إلى الزبون وفي حساب كود التحقق من صحتها وصحة مصدرها. يعتمد بروتوكول المصافحة على عدة أدوات لتشارك المفتاح السري بين المخدم والزبون، وتعد خوارزمية ديفي-هيلمان *Diffie-Hellman* الأكثر استخداماً.

4. يرد الزبون على المخدم بإرسال شهادته الرقمية في حال كانت مطلوبة منه، كما يؤكد على نجاح التحقق من الشهادة الرقمية الخاصة بالمخدم.

5. تتم عملية تبادل لمفتاح آخر سري مشترك بين المخدم والزبون، سيستخدمه الزبون لاحقاً في تعمية البيانات التي يرسلها إلى المخدم وفي حساب كود التحقق من صحتها وصحة مصدرها.

6. في النهاية يستخدم الطرفان بروتوكول تغيير محددات التعمية *SSL Change Cipher Spec Protocol* لاعتماد المفاتيح المنشأة وخوارزميات التعمية وغيرها من المعاملات المطلوبة لتنفيذ التعمية وحساب كود التحقق من الرسائل أثناء الجلسة الآمنة.



الشكل *(24)*: بروتوكول المصافحة

ملاحظة: تستخدم الشهادات الرقمية *digital certificates* في بروتوكول المصافحة *SSL Handshake Protocol* بشكل أساسي ليتحقق الطرفان من هويات بعضهما البعض *peer authentication*، أي أن المفتاح العمومي *public key* الذي تحمله شهادة رقمية لا يستخدم للتعمية اللامتناظرة في *SSL* لأن التعمية في هذه التقانة متناظرة. بالمقابل، قد يستخدم المفتاح العمومي لتبادل مفاتيح التعمية المتناظرة بشكل آمن بين الطرفين، وذلك تبعاً للتهيئة *configuration* المحددة لبروتوكول المصافحة *SSL Handshake Protocol*.

بروتوكول السجلات *SSL Record Protocol*

يهدف هذا البروتوكول إلى تقديم خدمات الأمن المطلوبة من طبقة المقابس الآمنة. لا ينتمي هذا البروتوكول إلى مستوى التطبيق *application layer* فهو لا يرتبط بتطبيق معين، إلا أنه لا ينتمي إلى مستوى النقل *transport layer* وإنما يعمل بين المستويين لحماية جلسة النقل *transport session* الخاصة بالتطبيق الذي يطلب استخدام *SSL*. يتبع هذا البروتوكول منهجية تغليف *encapsulation* البيانات القادمة من المستوى الأعلى لإنشاء طرود *packets* تمثل البيانات بالنسبة للمستوى الأدنى، وهذا ما يعطي تقانة *SSL* تسمية الطبقة *layer* في مكدس بروتوكولات الإنترنت *TCP / IP Stack* مهما اختلفت طريقة تنفيذ بروتوكولات *SSL*. يعمل بروتوكول السجلات *SSL Record Protocol* على الشكل التالي:

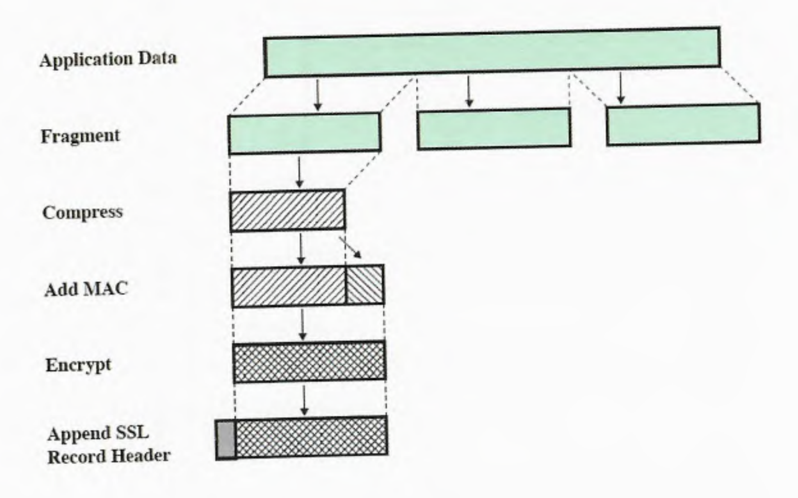
1. تقسيم بيانات التطبيق *application data* إلى قطع *fragment* ستشكل كل منها بيانات لطرد *SSL*.

2. إجراء عملية ضغط لبيانات الطرد.

3. حساب كود التحقق *MAC: Message Authentication Code* للطرد باستخدام المفتاح السري المشترك الذي تم توليده خلال تشغيل بروتوكول المصافحة *SSL Handshake Protocol* الخاص بالجلسة الحالية، ولصق كود التحقق بالطرد.

4. إجراء عملية تعمية متناظرة *symmetric encryption* للطرد مع كود التحقق الملصق به. وذلك باستخدام خوارزمية التعمية والمفتاح السري المتفق عليهما بنتيجة تنفيذ بروتوكول المصافحة *SSL Handshake Protocol* الخاص بالجلسة الحالية.

5. لصق ترويسة بروتوكول السجلات *SSL Record Header* بالبيانات المعماة لينتج عن ذلك طرد بروتوكول السجلات *SSL Record Packet* الذي سيشكل بيانات لطرد بروتوكول مستوى النقل *TCP*.



الشكل *(25)*: بروتوكول السجل

6 - 3 - بروتوكول الإنترنت الآمن *IPSec*

قد لا تفي الخدمات الأمنية على مستوى التطبيق *application layer* أو مستوى النقل *transport layer* بالغرض في أنظمة معينة تحتاج إلى توفير هذه الخدمات على مستوى الشبكة *network layer* ككل وليس لجلسة واحدة أو كتلة واحدة من البيانات. يقدم بروتوكول الإنترنت الآمن *IPSec* خدمات الأمن على مستوى الشبكة *network layer* بحيث تتحقق خواص سرية البيانات *data confidentiality* وسلامة البيانات *data integrity* والتحقق من مصدر البيانات *data source authentication* بشكل دائم في الشبكة مهما تكون الجلسة المفعلة على مستوى النقل *transport layer* ومهما يكون الإجراء الذي تتبع له هذه الجلسة على مستوى التطبيق *application layer*. يحقق بروتوكول الإنترنت الآمن سرية البيانات باستخدام التعمية المتناظرة *symmetric encryption*، ويحقق سلامة البيانات والتحقق من مصدر البيانات باستخدام خوارزمية *HMAC*. يعتمد بروتوكول الإنترنت الآمن على مفاتيح سرية يتم إنشاؤها خلال مرحلة بناء الشبكة الآمنة ولاحقاً كلما انضم طرف جديد إليها.

قدم مصممو النسخة السادسة من بروتوكول الإنترنت *IPv6* حلاً لتوفير خدمات الأمن على مستوى الشبكة من خلال تعريف ترويسة *header* خاصة بذلك يمكن إضافتها إلى الطرد *packet* عند الرغبة في استخدام هذا الحل الأمني، وتكون بيانات الطرد معماة وكود التحقق منها جزء من الترويسة. نشأ بروتوكول الإنترنت الآمن *IPSec* من تطبيق الحل الأمني المصمم لبروتوكول *IPv6* على النسخة الرابعة من بروتوكول الإنترنت *IPv4*. يقدم البروتوكول *IPv4* إمكانية إضافة أجزاء مخصصة إلى ترويسة الطرد *packet header* في مساحة الخيارات والحشو *options / padding*. ينشأ طرد البروتوكول *IPSec* من استخدام مساحة الخيارات والحشو في البروتوكول *IPv4* لإضافة أجزاء

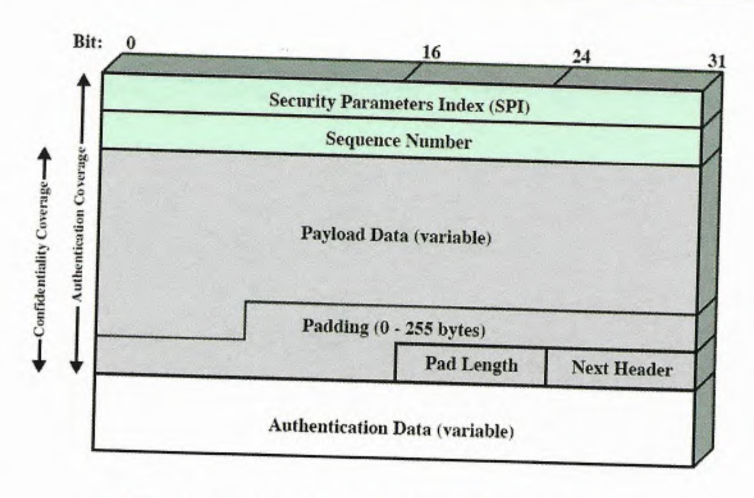
تماثل ترويسة الأمن *security header* في البروتوكول *IPv6*، بالإضافة إلى تعمية بيانات الطرد. قدمت معايير البروتوكول *IPSec* تباعاً العديد من الأنماط لبناء الطرد الآمن *IPSec Packet*، واستقرت في نسخها الأخيرة على النمط المسمى محتوى الأمن المغلف *ESP: Encapsulated Security Payload*، حيث يتألف الطرد في هذا النمط، بالإضافة إلى الترويسة المعيارية للبروتوكول *IPv4*، من الأجزاء التالية:

1. مؤشر معاملات الأمن *SPI: Security Parameters Index*. تمثل القيمة الموجودة في هذا الحقل محدداً لسجل المرسل في قاعدة البيانات الأمنية لدى المرسل إليه. تنشأ قاعدة البيانات الأمنية لدى كل طرف في مرحلة الترابط الأمني *SA: Security Association*، وهي مرحلة تحضيرية لبناء الشبكة الآمنة بالاعتماد على البرتوكول *IPSec* ويتم فيها تشارك المفاتيح السرية والاتفاق على أدوات وخوارزميات التعمية لتي يخزنها كل طرف على شكل سجلات يخص كل منها واحداً من الأطراف الأخرى في الشبكة الآمنة.

2. رقم تسلسلي *sequence number*.

3. البيانات المعماة *encrypted payload*، مع حشو *padding* لإتمام حجمها إلى مضاعفات 32 بت، مع حقل يعطي طول الحشو *pad length*، وأخيراً حقل موجود بسبب استخدام الحل الأمني الخاص بالبروتوكول *IPv6* دون تغيير ويسمى هذا الحقل الترويسة التالية *next header*.

4. بيانات التحقق *authentication data* والمقصود التحقق من سلامة بيانات الطرد وهوية مصدره، فهي ناتج تطبيق خوارزمية *HMAC* على الحقول السابقة في الطرد باستخدام مفتاح سري مشترك بين المرسل والمرسل إليه تمت مشاركته بشكل آمن في مرحلة الترابط الأمني *SA: Security Association*.



الشكل (26): تصميم الطرد من نمط *ESP* في البروتوكول *IPSec*

ملاحظة: عندما نحافظ على ترويسة *IPv4* كما هي وننشئ ونضيف طرد *IPSec* من نمط *ESP* المشروح أعلاه يكون نمط العمل في الشبكة الآمنة الناتجة هو نمط النقل *Trnasport Mode*. ويفيد في تقديم الخدمات الأمنية المشروحة أعلاه لكافة جلسات النقل *transport sessions* في الشبكة، ويحافظ على عناوين الإنترنت *IP* الخاصة بالأطراف المختلفة بدون تعمية لتتم عملية التسيير *routing* بالشكل المطلوب. أما عندما نعتبر طرد *IPv4* بكامله بيانات بحاجة للتعمية، وننشئ الحقول المطلوبة لتشكيل طرد *IPSec* من النمط *ESP* المشروح أعلاه، ومن ثم نضيف إلى بداية الطرد ترويسة *IPv4* جديدة تحمل عنوان مخدم مخصص يعمل كوسيط *proxy*، يكون نمط العمل في الشبكة الآمنة الناتجة هو نمط النفق *Tunnel Mode*. ويفيد في إنشاء الشبكات الافتراضية الخاصة *VPN: Virtual Private Networks*، حيث تعتمد خصوصيتها على إخفاء معلومات الطرود كاملةً بما فيها عناوين الإنترنت *IP addresses* لأطراف الاتصال في الشبكة الآمنة.

الفصل السابع

الأمن الفيزيائي *Physical Security*

أمن المعلومات يشمل حماية المعلومات ومكونات نظام المعلومات *(كونها ممتلكات)* والعاملين على نظام المعلومات إضافة إلى مواقع تواجدها ومنع الوصول إليها *(الأمن الفيزيائي)*.

يمكن الحصول على المعلومات بسهولة أكبر إذا تمكن المتطفل *Intruder* من دخول موقع نظام المعلومات مثل غرفة الحواسيب أو الخدمات، أو غرفة تجهيزات الاتصالات، أو غرفة المقاسم الهاتفية بحيث يتمكن من الوصول إلى أقراص تخزين المعلومات *(الأقراص الصلبة HDs أو CDs أو الأشرطة المغناطيسية tapes)* أو يتمكن من زرع تجهيزات التقاط المعلومات ليحصل على المعلومات بطريقة أو بأخرى *(إعادة إرسالها)*. كما يمكن للمتطفل إخراج نظام المعلومات من الخدمة إذا تمكن من الوصول إليه وتخريبه أو تعطيل أحد مكوناته.

استعرضنا في الفصول السابقة مواضيع الأمن المنطقي *logical security* التي تركز على حماية البيانات والشبكات وسنتطرق هنا إلى مواضيع الأمن الفيزيائي *physical security* الذي تمنع الضرر أو سوء الاستخدام للبنية التحتية الفيزيائية وتتضمن:

- حماية التجهيزات الحاسوبية ووسائط التخزين من التخريب أو سوء الاستخدام أو السرقة
- حماية الأشخاص المشغلين للأنظمة الحاسوبية
- منع الدخول لمواقع التجهيزات واتخاذ الإجراءات التي تتعلق بحماية البناء الحاوي للأجهزة
- اتخاذ إجراءات الحماية من الحريق والحماية البيئية.

7 - 1 - مهددات الأمن الفيزيائي

تتضمن مهددات الامن الفيزيائي ثلاثة أنواع رئيسية:

- **المهددات البيئية *environmental threats***
  - الخدمات مصممة لتعمل في درجات حرارة محيطية ضمن المجال *10-32* درجة مئوية وبالتالي تؤدي الحرارة المرتفعة والبرودة المنخفضة والرطوبة العالية إلى سوء في أداء التجهيزات وربما تؤدي لتوقفها عن العمل. ولتلافي ذلك توضع مكيفات (سخن-بارد) في غرفة الخدم للحفاظ على درجات حرارة ضمن المجال المذكور.
  - الحريق والدخان: يجب وضع كواشف دخان واتخاذ الإجراءات لمكافحة الحريق
  - الغبار والسوائل (مثل المياه والمواد الكيميائية)، العفن والحشرات والقوارض.
  - الكوارث الطبيعية مثل الفيضان والزلزال والعواصف الثلجية وغيرها
- **المهددات التقنية**
  - مشاكل التغذية الكهربائية مثل ارتفاع وانخفاض الجهد الكهربائي والضجيج على خطوط التغذية الكهربائية ويمكن التغلب عليها باستخدام وحدات التغذية عديمة الانقطاع *UPS* ومولدات التيار الكهربائي.
  - التداخل الكهرطيسي من التجهيزات المجاورة مثل المحركات والمراوح والتجهيزات الثقيلة ومحطات الاتصالات الراديوية والميكروية القريبة والتي تسبب تقطعات في عمل التجهيزات الحاسوبية. ويتم التغلب عليها بتحجيب غرف الخدمات (استخدام مايسمى قفص فرادي).

- **المهددات التي يسببها الإنسان**
  - الدخول غير المصرح به لموقع التجهيزات والذي قد يسبب تخريب أوسرقة التجهيزات ووحدات التخزين أو سوء استخدامها. ويتم التغلب عليها بقفل أبواب غرف التجهيزات الحاسوبية واستخدام تجهيزات التحكم بالدخول عن طريق البصمة او القزحية فقط للأشخاص المخولين بالدخول إليها.

7 - 2 - التعافي من خروقات الأمن الفيزيائي

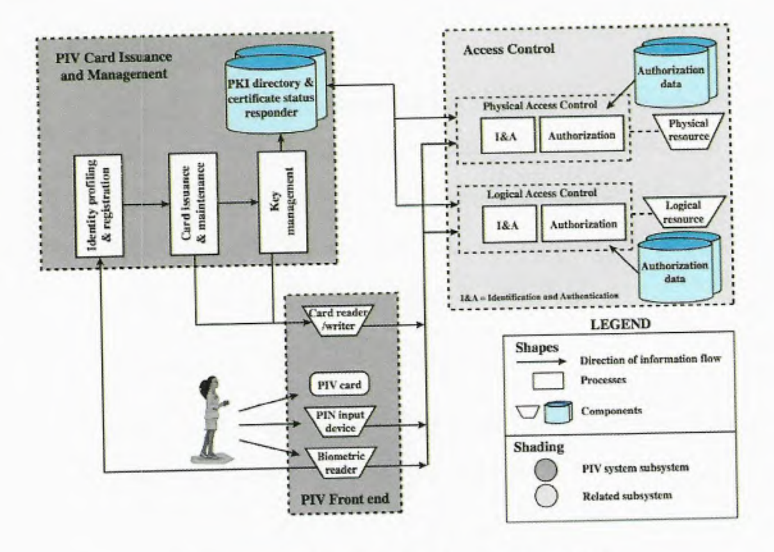
يتم اتخاذ إجراءات للتعافي من خروقات الأمن الفيزيائي مثل التخزين الاحتياطي للمعلومات *backup* أو تكرار التجهيزات *redundancy* من خلال استضافة مخدمات احتياطية في مراكز المعلومات *data centers* التي توفر كل إجراءات الحماية الفيزيائية والمنطقية للتجهيزات الحاسوبية.

7 - 3 - التكامل بين الأمن الفيزيائي والمنطقي

المقصود بالتكامل هنا تكامل وسائل التحقق من المستخدم *personal identity verification PIV*

للدخول إلى الموقع الفيزيائي للمخدمات والدخول المنطقي للنظم الحاسوبية. يوضح الشكل *(27)* نظام موحد

للتحقق من الشخصية يستخدم للدخول للموقع الفيزيائي وللنظام الحاسوبي. يتكون النظام الموحد من ثلاثة مكونات أساسية هي:



الشكل (27) نظام موحد للتحقق من الشخصية *PIV*

- نظام إصدار وإدارة للبطاقات *PIV* ويتضمن مايلزم لإصدار بطاقات ذكية تتضمن بيانات المستخدم الشخصية ومميزاته البيومترية أو الحيوية (بصمة مثلاً) مشفرة باستخدام نظام التشفير بالمفتاح العمومي إضافة إلى ما يلزم لتوليد وإدارة مفاتيح التشفير وإصدار شهادة رقمية.
- واجهة الدخول الأمامية *PIV front end* للموقع وللنظام الحاسوبي تتضمن قارئ بطاقات ذكية وقارئ بصمة حي (كمميزة بيومترية).
- نظام دخول للموقع وللنظام الحاسوبي *access control* ويتضمن نظام معالجة للتحقق من المميزات البيومترية للمستخدم ومطابقتها مع تلك المخزنة في النظام الحاسوبي وذلك لحالتي دخول للموقع والدخول للنظام الحاسوبي.

الفصل الثامن

إدارة الأمن- المعايير *Security Management - Standards*

8 - 1 - تعريف إدارة أمن النظم المعلوماتية

إدارة أمن النظم المعلوماتية هي عملية تستخدم لتحقيق والحفاظ على مستويات مناسبة من السرية *confidentiality* وسلامة المعلومات *integrity* والإتاحية *availability* والتحقق *authenticity* والمسألة *accountability* والموثوقية *reliability*.

تعمل إدارة أمن النظم المعلوماتية للإجابة عن الأسئلة التالية:

- ماهي الممتلكات التي نريد حمايتها ؟
- كيف يتم تهديد هذه الممتلكات؟
- ماذا يمكن أن نعمل لكي نواجه أو نعاكس هذه المهددات ؟

وتقدم الحلول اللازمة بطريقة فعالة من حيث الكلفة *in cost-effective manner*

يتم في عملية إدارة الامن تقدير المخاطر *risk assessment* الأمنية لكل الممتلكات التي تحتاج لحماية في المؤسسة بحيث تجيب عن الأسئلة الثلاثة أعلاه

8 - 2 - وظائف إدارة الأمن

- تحديد الأهداف الأمنية والأخطار بشكل عام (من خلال وضع سياسة أمنية *security policy*)
- تحديد حجم الخطر لكل الممتلكات التي تحتاج حماية
- تحديد الحمايات وتنفيذها *implementation*
- مراقبة عمل وصيانة الحمايات للتأكد من تحقيقها للأهداف المرجوة *check and maintain*
- تكرار كامل العملية مع الزمن نظراً للتغير السريع في التقانات والأخطار

عملية تقدير الأخطار تقدم المعلومات اللازمة حول الإجراءات الإدارية والتشغيلية والفنية اللازمة لتقليل الأخطار المعروفة لمستوى مقبول، أو التخلص منها، أو توفير إجراءات تخفيفة *mitigating controls*

8 - 3 - معايير إدارة أمن النظم المعلوماتية

تقدم سلسلة المعايير *ISO27001-ISO27005* إجرائيات لإدارة أمن النظم المعلوماتية نذكر منها:

- *ISO27001* توصيف تنفيذ الإجراءات الأمنية وصيانتها بهدف الحصول على شهادة *ISMS information security management system* العالمية لاعتمادية إجرائيات الأمن.
- *ISO27002* تركز على طيف واسع من الإجراءات الأمنية مثل إدارة المخاطر السياسة الأمنية، الأمن الفيزيائي، أمن الاتصالات، إجرائيات التحكم بالدخول، إدارة الحوادث والأزمات، إدارة التعافي من الأزمات وتحقيق استمرارية العمل.... الخ.

8 - 4 - السياسة الأمنية

حماية النظم المعلوماتية ترتكز على ثلاثة عوامل: السياسة الأمنية، الحلول التقنية للأمن، إجراءات وإرشادات للعنصر البشري.

السياسة الأمنية للمعلومات: مجموعة القواعد والتعليمات الناظمة التي تضعها إدارة المؤسسة ليطبقها العاملون لدى التعامل مع نظم المعلومات (بكافة أشكالها) داخل المؤسسة وتتصل بشؤون الدخول الى المعلومات والعمل على نظمها وادارتها. وتهدف السياس الأمنية إلى:

- تعريف المستخدمين والاداريين بالتزاماتهم وواجباتهم المطلوبة لحماية نظم الحواسيب والشبكات وكذلك حماية المعلومات بكافة أشكالها. وفي مراحل ادخالها ومعالجتها وخزنها ونقلها واعادة استرجاعها
- تحديد التكنولوجيات التي يتم من خلالها تحقيق وتنفيذ الواجبات المحددة على كل من له علاقة بالمعلومات ونظمها وتحديد المسؤوليات عند حصول الخطر.
- بيان الإجراءات المتبعة لتجاوز التهديدات والمخاطر والتعامل معها والجهات المناط بها القيام بها بذلك.

تحدد الإدارة ثلاثة أنواع من السياسات الأمنية:

- سياسة أمنية عامة تضعها إدارة المؤسسة تحدد فيها الهدف من هذه السياسة والمسؤوليات والعقوبات والإجراءات التأديبية عند خرق سياسة أمن المعلومات.

- سياسة أمنية تتعلق بنوع من مكونات النظام المعلوماتي مثلاً استخدام الحواسيب الشخصية، استخدام الانترنت والتصفح، كيفية استخدام تجهيزات الشبكة والاتصالات، كيفية منع الاختراق، التعامل مع مضادات الفيروسات، سياسة البريد الإلكتروني، سياسة كلمات المرور، الامن الفيزيائي ... الخ. ويتم وضع وثيقة خاصة بكل نوع.

- سياسة أمنية تفصيلية لكل جهاز ضمن كل نوع من مكونات النظام المعلوماتي تحدد فيها طريقة وصله وبرمجته. مثلاً لجدار ناري *firewall* طريقة ربطه ضمن الشبكة المعلوماتية وبرمجة لائحة المرور فيه *access control* وفق القواعد التي تحددها إدارة المنظومة الامنية.

السياسة الأمنية الشاملة هي تجميع للسياسات الأمنية الفرعية لمختلف لكل مكون من مكونات النظام المعلوماتي.

المخطط الأزرق للأمن

انطلاقاً من وثائق السياسة الأمنية يتم وضع مخطط تصميمي يسمى المخطط الأزرق للأمن *security blueprint* يتضمن التصاميم التفصيلية لمكونات المنظومة الأمنية وكيفية تنفيذها والتجهيزات والبرامج المستخدمة للحماية، إضافة إلى برامج التدريب وتأهيل الفنيين العاملين على التجهيزات الأمنية وبرامج التوعية لكل مستخدمي المنظومة المعلوماتية في المؤسسة وإجرائيات وبرامج الصيانة الدورية.

8 - 5 - التخطيط للطوارئ

التخطيط للطوارئ *contingency planning* هو التخطيط الكامل للتحضير للاستجابة والتعافي من الحوادث التي تهدد أمن النظم المعلوماتية وكيفية العودة للعمل الطبيعي للمنظومة المعلوماتية. يوجد ثلاثة أنواع من الخطط:

- خطط الاستجابة للحوادث *(incident response plans (IRPs)* تركز على الاستجابة الفورية لكن إذا تصاعد الهجوم يتم التحول إلى إجرائية التعافي من الكارثة والوصول لاستمرار عمل المنظومة.

- تركز خطط التعافي من الكوارث *disaster recovery plans (DRPs)* عادة على استعادة المنظومة بعد وقوع الكوارث، والوصول لاستمرار عمل المنظومة.

- خطط استمرارية العمل *business continuity plans (BCPs)* تتم بالتزامن مع خطط التعافي من الكوارث عندما تكون الأضرار كبيرة أو طويلة الاجل ما يتطلب أكثر من مجرد استعادة المعلومات وعمل المنظومة المعلوماتية وتتطلب إجرائيات غير بسيطة لاستعادة العمل.

الفصل التاسع

الجرائم المعلوماتية *E-crimes*

9 - 1 - تعريف الجريمة المعلوماتية

جريمة تُرتكب باستخدام الأجهزة الحاسوبيّة أو الشّبكة المعلوماتية، أو تقع على المنظومات المعلوماتيّة أو الشّبكة.

9 - 2 - الجرائم المعلوماتية التقليدية

الجرائم المرتكبة عبر الشبكة المعلوماتية او الانترنت (بريد الكتروني، صفحات إعلانية عبر الانترنت،...الخ) أو الوسائل الالكترونية والمنصوص عنها في القوانين الجزائية مثل:

- التهديد والابتزاز
- الذم والقدح
- المساس بالآداب والاخلاق العامة
- تسهيل الدعارة أو الفجور
- الاساءة للمقدسات والشعائر الدينية
- الترويج والاتجار بالمخدرات
- غسيل الأموال
- الترويج للأعمال الإرهابية

هذه الجرائم تقليدية وموصفة في قانون العقوبات السوري وفي القوانين الجزائية الأخرى الخاصة بكل جريمة. نص قانون العقوبات على التشدد في عقوبة بعض الجرائم إذا تمت باستخدام وسائل العلنية وتعتبر الشبكات والانترنت من الوسائل العلنية.

9 - 3 - الجرائم المعلوماتية المستحدثة

الجرائم المرتكبة عبر الشبكة المعلوماتية او الانترنت (بريد الكتروني، صفحات إعلانية عبر الانترنت،...الخ) أو الوسائل الالكترونية المستحدثة مع دخول الإنترنت والشبكات مثل:

- الدخول غير المشروع إلى منظومة معلوماتية
- شغل اسم موقع الكتروني
- إعاقة الوصول إلى الخدمة
- اعتراض المعلومات (تنصت)
- تصميم البرمجيات الخبيثة واستخدامها
- إرسال البريد الواغل *maps*
- الاحتيال عن طريق الشبكة
- الاستعمال غير المشروع للبطاقات المصرفية
- انتهاك حرمة الحياة الخاصة

9 - 4 - قانون "تنظيم التواصل على الشبكة ومكافحة الجريمة المعلوماتية"

صدر المرسوم *17* لعام *2012* المتضمن قانون "تنظيم التّواصل على الشّبكة ومكافحة الجريمة المعلوماتيّة" وتضمن خمسة فصول تتضمن تعاريف وفقرات خاصة بتنظيم التواصل على الشبكة ومكافحة الجرائم المعلوماتية والضابطة العدلية والدليل الرقمي وأحكام عامة وختامية.

عرف القانون جملة من المصطلحات المعلوماتية نذكر أهمها:

مقدم الخدمات على الشبكة: أيّ من مقدّمي الخدمات الذين يعملون في إطار التّواصل على الشّبكة؛ ومن أصنافهم: مقدّم خدمات النّفاذ إلى الشّبكة، ومقدّم خدمات التّواصل على الشّبكة، ومقدّم خدمات الاستضافة على الشّبكة.

بيانات الحركة: أيّ معلومات يجري تداولها في إطار التّواصل على الشّبكة تحدّد، بوجه خاص، مصدر الاتّصال ووجهته ومساره والمواقع الإلكترونيّة التي يجري الدّخول إليها ووقت الاتّصال ومدّته.

التقصّي الإلكتروني: الوصول، المأذون به قانوناً، إلى المعلومات أو بيانات الحركة المتداولة على المنظومات المعلوماتيّة أو الشّبكة لأغراض التعقّب أو الضّبط أو التّحقيق. وتقوم به الضابطة العدلية بناءً على إذن قضائي.

الدّليل الرقمي: البيانات الرّقمية المخزّنة في الأجهزة الحاسوبيّة أو المنظومات المعلوماتيّة، أو المنقولة بواسطتها، والتي يمكن استخدامها في إثبات أو نفي جريمة معلوماتيّة. وتعتبر الأدلة الرقمية أدلة ثابتة يأخذ بها القضاء مالم يثبت تزويرها.

الضابطة العدلية: أتاح للضابطة العدلية تفتيش وضبط الحواسيب والبرمجيات والبيانات

حدد القانون مسؤوليات عامة تقع على عاتق مقدمي الخدمات على الشبكة:

- حفظ نسخة من المحتوى المخزّن لديهم، في حال وجوده، وحفظ بيانات الحركة التي تسمح بالتّحقّق من هوية الأشخاص الذين يسهمون في وضع المحتوى على الشّبكة، وذلك لمدّة تحدّدها الهيئة. وتخضع هذه البيانات والمحتوى لسرّ المهنة. وتضع الهيئة النّواظم والمعايير

التّقنيّة اللازمة لتطبيق أحكام هذه الفقرة، بعد التّنسيق مع المجلس الوطني للإعلام في ما يخصّ وسائل التّواصل على الشّبكة، وفق ما ينصّ عليه قانون الإعلام النّافذ.

- تقديم أيّ معلومات تطلبها منهم السّلطات القضائيّة المختصّة.
- التّعريف عن الموقع الإلكتروني لمقدّم خدمات التّواصل على الشّبكة
- الإخبار عن الطّابع غير المشروع لمحتوى على الشّبكة
- حجب موقع إلكتروني
- الإخلال بالالتزام بحفظ نسخة من المحتوى وبيانات الحركة
- إفشاء البيانات والمعلومات
- تغيير المحتوى
- الامتناع عن إجابة أمر السّلطة القضائيّة
- الامتناع عن حذف محتوى غير مشروع أو تعديله أو تصحيحه
- تطبيق قانون التّجارة
- مراعاة حقوق المؤلّف والملكيّة

واعتبر القانون النطاق العلوي السوري *(sy.)* في حكم الأراضي السورية وتطبق عليه القوانين السورية.

كما حدد القانون غرامات مالية وعقوبة الحبس على كل جريمة من الجرائم المعلوماتية المستحدثة بينما ضاعف العقوبة المنصوص عنها في القوانين الجزائية للجرائم التقليدية إذا ارتكبت عبر الشبكة والانترنت.

الفصل العاشر

الأسئلة العامة

الفصل الأول: مدخل إلى أمن المعلومات والنظم المعلوماتية

1. أمن نظم المعلومات هو حماية نظم المعلومات ومكوناتها الهامة من:

a. خطر الحريق.

b. قوى الطبيعة.

c. الاقتحام والسرقة.

***d*. <u>كافة الأخطار.</u>**

2. لحماية المعلومات ونظمها ومؤسساتها، يجب علينا استخدام ما يلي:

a. سياسة حماية المعلومات.

b. التدريب على إدارة الأمن.

c. توعية المدراء والكادر المعلوماتي.

***d*. <u>سياسة وتوعية وتدريب وتقانات.</u>**

3. الضعف *vulnerability* هو:

a. إمكانية مواجهة البرمجيات الخبيثة.

b. إمكانية مواجهة أي هجوم.

***c*. <u>ثغرة بسبب حمايات غير مطبقة أو غير فعالة.</u>**

d. التهديدات الأخطر للنظم الحاسوبية.

4. أحد اهداف أمن المعلومات توفير المعلومات والتطبيقات عند طلبها لتمكن النظام المعلوماتي من أداء دوره يدعى:

a. الوثوقية *reliability*.

b. سلامة المعلومات *integrity*.

c. **الاتاحية *availability*.**

d. الاستيقان *authenticity*.

5. أحد اهداف أمن المعلومات توفير الإجراءات التي تمكننا من تحديد المسؤوليات عند القيام بعمل باستخدام النظام المعلوماتي يدعى:

a. **المساءلة *accountability*.**

b. السرية *confidentiality*.

c. الاتاحية *availability*.

d. الاستيقان *authenticity*.

الفصل الثاني: أدوات التعمية

1. عندما يحاول مهاجم تجربة كافة مفاتيح التشفير الممكنة في خوارزمية فك التشفير لفك الرسالة المشفرة والوصول للرسالة الاصلية الفعلية تسمى هذه الطريقة:

a. **هجوم القوة الغاشمة *brute force*.**

b. كسر الشيفرة.

c. هجوم الرجل في المنتصف.

d. هجوم حجب الخدمة.

2. طريقة التشفير بالمفتاح الوحيد *OTP* تستخدم:

a. مفتاح سري طول *2048* بت.

b. مفتاح شبه عشوائي بطول *256* بت.

c. مفتاح سري بطول *1024* بت.

***d.* مفتاح عشوائي بطول الرسالة.**

3. المشفرات التي تستخدم عملية *XOR* بين خرج المولد شبه العشوائي والبيانات تدعى:

a. المشفرات المتناظرة الكتلية.

b. توابع التهشير.*hash functions*

***c.* المشفرات المتناظرة التسلسلية.**

d. المشفرات اللامتناظرة.

4. عند تعمية رسالة بمفتاح سري متناظر، ثم تعمية هذا المفتاح باستخدام المفتاح العمومي للمرسل إليه ولصق الناتج بالرسالة المعماة، نكون قد حصلنا على:

a. توقيع رقمي.

***b.* ظرف رقمي.**

c. شهادة رقمية.

d. هوية رقمية.

5. ميزة موجودة في التوقيع الرقمي *digital signature* وغير موجودة في خوارزمية *MAC*

a. سلامة البيانات *data integrity*.

b. التحقق من هوية المرسل.

c. **منع الإنكار *non-repudiation*.**

d. الإتاحية *availability*.

6. لكسر سرية بروتوكول ديفي-هيلمان *Diffie-Hellman* وكشف مفاتيح التشفير المتبادلة بين الطرفين قد يحتاج المهاجم إلى عملية رياضية (تعتبر غير ممكنة حسابياً):

a. **اللوغاريتم المتقطع *discrete logarithm*.**

b. تحليل عدد ضخم إلى عوامله الأولية.

c. لوغاريتم عدد ضخم.

d. التحليل المتقطع للعوامل الأولية.

7. سينتج عن تشفير كتلتين متطابقتين في النص الواضح *plaintext* كتلتان مختلفتان في النص المشفر *ciphertext* باستخدام:

a. بروتوكول *Diffie-Hellman*.

b. ***Cipher Block Chaining (CBC)*.**

c. خوارزمية *MAC*.

d. خوارزمية *RSA*.

8. يمكن استخدام خوارزمية *RSA* لتحقيق ما يلي:

a. السرية والتحقق من الهوية.

b. السرية فقط.

c. تشارك المفاتيح فقط.

***d.* <u>السرية والتحقق من الهوية وتشارك المفاتيح.</u>**

9. تسمح التعمية المتناظرة *symmetric encryption* بإخفاء البيانات أثناء نقلها، ولكنها لا تستطيع أن تضمن ما يلي:

a. خاصة السرية *Confidentiality*.

b. خصوصية البيانات *Data Privacy*.

c. التحقق من الهوية *authentication*.

***d.* <u>منع الإنكار *non-repudiation*.</u>**

10. يؤدي نجاح فك تعمية التوقيع الرقمي *digital signature* باستخدام المفتاح العمومي *public key* الصحيح إلى التحقق من:

a. سلامة البيانات *data integrity*.

***b.* <u>منع الإنكار *non-repudiation*.</u>**

c. التحقق من هوية المصدر.

d. ورود الرسالة بالترتيب الصحيح.

11. بالنسبة لأي تابع تهشير *hash function*، يجب أن تكون العملية التالية غير ممكنة حسابياً *computationally infeasible*:

a. صياغة توقيع رقمي.

***b*. إيجاد رسالتين لهما قيمة التهشير ذاتها.**

c. تعمية قيمة التهشير.

d. التحقق من سلامة البيانات.

12. إذا أردت استخدام خوارزمية *RSA* للتحقق من هوية مصدر رسالة وصلتك، فأنت بحاجة لاستخدام:

a. مفتاحك الخاص.

b. مفتاحك العمومي.

***c*. المفتاح العمومي للمرسل.**

d. مفتاح التطبيق العمومي.

13. من أجل توليد التوقيع الرقمي *digital signature* لرسالة ما اعتماداً على التعمية اللامتناظرة نقوم بما يلي:

a. تعمية الرسالة بخوارزمية *AES*.

b. تعمية الرسالة بالمفتاح الخاص للمرسل.

c. تعمية هاش الرسالة بخوارزمية *DES*.

***d*. تعمية هاش الرسالة بالمفتاح الخاص للمرسل.**

14. عند توليد ظرف رقمي *digital envelope* نقوم بتعمية الرسالة باستخدام خوارزمية تستخدم:

a. مفتاح خاص *private key* لامتناظر.

b. مفتاح عام *public key*.

c. **مفتاح سري متناظر.**

d. تهشير *hashing*.

15. تستخدم الشهادات الرقمية *digital certificates* للتحقق من الربط بين:

a. زوج من المفاتيح اللامتناظرة.

b. مفتاح سري وخوارزمية تعمية.

c. **المفتاح العمومي وهوية مالكه.**

d. رسالة وقيمة تهشير *hash value*.

16. يعتمد التحقق من شهادة رقمية *digital certificate* على التحقق من التوقيع الرقمي *digital signature* الذي يخص:

a. مالك الشهادة.

b. **سلطة إصدار الشهادة.**

c. مستلم الرسالة.

d. صاحب التوقيع الرقمي.

17. عندما تستقبل بريداً إلكترونياً موقعاً رقمياً، وتتحقق من أن التوقيع صحيح، يمكنك أن تكون متأكداً مما يلي:

a. لم يقرأه أحد غيرك أثناء انتقاله.

b. لم يستقبله أحد غيرك.

c. لم تتم سرقة الرسالة.

d. صحة الرسالة وهوية المرسل.

18. تعتبر خوارزمية AES أكثر أماناً من خوارزمية DES للسبب التالي:

a. تستخدم AES مفاتيح قصيرة.

b. تستخدم DES مفاتيح أطول.

c. تستخدم AES مفاتيح أطول.

d. تستخدم AES الشهادات الرقمية.

19. تمكن خوارزمية التحقق من الرسائل MAC من:

a. التحقق من سلامة الرسالة فقط.

b. التحقق من هوية المستخدم فقط.

c. التحقق من سلامة الرسالة وهوية المرسل.

d. التحقق من سرية الرسالة.

20. تمكن خوارزمية *HMAC* من:

a. **التحقق من سلامة الرسالة ومصدرها.**

b. التحقق من سرية الرسالة.

c. التحقق من وجهة الرسالة.

d. التحقق من مصدر الرسالة.

الفصل الثالث: التحقق من المستخدم

1. من تقنيات أمن كلمات السر المنتشرة:

a. استخدام التشفير وفك التشفير.

b. **استخدام التهشير مع قيمة ملحية *salt*.**

c. استخدام تشفير *MD5*.

d. استخدام تشفير *DES*.

2. تقنية كلمة المرور المهشرة تكون أكثر مناعة للكسر في حال:

a. استخدام قيمة ملحية *salt 8* بت.

b. استخدام قيمة ملحية *salt 12* بت وكلمة سر طويلة.

c. استخدام قيمة ملحية *salt 24* بت.

d. **استخدام قيمة ملحية *48* بت وكلمة سر طويلة.**

3. يتم التحقق من المستخدم عن بعد:

a. استخدام خوارزمية كلمة المرور المهشرة.

b. استخدام خوارزمية DES لتشفير كلمة المرور.

c. استخدام خوارزمية RSA لتشفير كلمة المرور.

d. استخدام بروتوكول challenge-response.

4. لاتستخدم للتحقق من هوية المستخدم بالنظم المعلوماتية:

a. كلمة المرور password.

b. المزايا البيومترية للمستخدم.

c. الهوية الشخصية.

d. البصمة مع بطاقة مغناطيسية.

الفصل الرابع: مهددات النظم المعلوماتية

1. لاتعتبر من مهددات النظم المعلوماتية:

a. الأخطاء البشرية.

b. الاختراق المتعمد.

c. تحديث مضاد الفيروسات.

d. قوى الطبيعة كالرياح والبرق.

2. تشكل الحلقة الاخطر على النظم المعلوماتية:

a. البرمجيات الخبيثة التي تهاجم النظم.

***b.* الاخطار الناجمة عن الموظفين في النظم.**

c. هجوم الحرمان من الخدمة DoS.

d. كسر كلمة المرور.

3. لا تعتبر من مؤشرات الإصابة بالفيروسات:

a. بطئ شديد في عمل الجهاز، وتباطئ في تنفيذ الأوامر.

b. بعض البرامج لا تعمل بشكل صحيح.

c. تظهر رسائل خطأ غير معروفة ومربعات حوار غير مألوفة.

***d.* كسر كلمة المرور.**

الفصل الخامس: أمن الشبكات

1. الجدار الناري يقوم:

***a.* التحكم بحركة البيانات بين الشبكة الداخلية والانترنت.**

b. منع دخول الفيروسات للشبكة الداخلية.

c. يمنع الاتصال بين الشبكة الداخلية وشبكة الانترنت.

d. يمنع إرسال الفيروسات إلى الشبكة البعيدة.

2. **الجدار الناري المرشح للطرود *packet filter firewall*:**

a. يفحص ترويسة ومحتوى الطرد *packet*.

b. يفحص محتوى الطرد *packet*.

c. <u>يفحص ترويسة الطرد.*packet*</u>

d. لا يفحص ترويسة الطرد *packet*.

3. **من نقاط ضعف لجدار الناري المرشح للطرود *packet filter firewall*:**

a. لا يقوم بفحص ترويسة كل الطرود الواردة عبره.

b. <u>لا يمكنه منع الهجوم الناتج عن البرمجيات الخبيثة.</u>

c. يمكنه منع الهجوم الناتج عن البرمجيات الخبيثة.

d. يقوم بترشيح الطرد استناداً إلى محتواه من بيانات.

4. نظام منع الاختراق *IPS* يفحص الطرود *packets* للتعرف على أي هجوم باستخدام تقنية:

a. الكشف على ترويسة الطرد *packet header* ومقارنتها مع بصمة البرنامج الخبيث.

b. <u>الكشف الاستدلالي أو المبني على التوقيع *signature/heuristic* للبرنامج الخبيث.</u>

c. مقارنة بين محتوى الطرود المتتابعة مع بعضها للكشف عن البرامج الخبيثة.

d. مقارنة ترويسة الطرد مع محتواه للكشف عن البرمجيات الخبيثة.

1. كم عدد المفاتيح التي يتم توليدها في بروتوكول المصافحة *handshake protocol* الخاص بطبقة المقابس الآمنة *Secure Sockets Layer (SSL)*؟

a. مفتاح عام مشترك وحيد.

b. مفتاحان عامان مختلفان.

c. مفتاح سري مشترك وحيد.

d. مفتاحان سريان مختلفان.

2. ينتمي البروتوكول التالي لمجموعة بروتوكولات طبقة المقابس الآمنة *SSL: Secure Sockets Layer*، ويعمل تحت مستوى التطبيقات *Application Layer*:

a. بروتوكول *HTTP*.

b. بروتوكول المصافحة *Handshake*.

c. بروتوكول السجلات *Record*.

d. بروتوكول الإنذار *Alert*.

3. يتم استخدام التقنية التالية للتحقق من الرسائل *message authentication* في بروتوكول الإنترنت الآمن *IPSec*:

a. القياسات الحيوية *biometrics*.

b. خوارزمية *HMAC*.

c. التوقيع الرقمي *digital signature*.

d. اسم مستخدم مع كلمة سر.

4. يمكن استخدام بروتوكول الإنترنت الآمن *IPSec* لإنشاء الشبكات الخاصة الافتراضية *VPN: Virtual Private Networks* لأنه:

a. **بروتوكول أمن يعمل في الطبقة الثالثة.**

b. تنفيذ للبروتوكول *IPV6*.

c. يرتبط بتطبيق خاص.

d. آلة افتراضية *virtual machine*.

5. يمكن استخدام البروتوكول *S / MIME* من أجل:

a. الولوج الآمن للشبكة عن بعد.

b. وصل فروع الشركة بشكل آمن.

c. **أمن البريد الإلكتروني.**

d. *VPN* بين المستخدم والشبكة

6. في بروتوكول الإنترنت الآمن *IPSec* يحتوي الطرد *packet* في النمط *ESP: Encapsulated Security Payload* على حقل يسمى *SPI: Security Parameters Index*، وذلك من أجل:

a. التحقق من الرسائل.

b. التحقق من مصدر البيانات.

c. سلامة البيانات.

d. **تحديد هوية العقدة.**

7. **طبقة المقابس الآمنة *Secure Sockets Layer (SSL)* هي:**

a. <u>**خدمة أمن على مستوى النقل *Transport*.**</u>

b. نظام كشف اختراقات الشبكة.

c. نمط من تطبيقات *IPSec*.

d. أداة تعمية في مستوى التطبيقات.

8. **يقدم بروتوكول السجلات *SSL Record Protocol* الخدمة التالية بالإضافة إلى التعمية:**

a. حجب الخدمة *Denial of Service*.

b. مراقبة المستخدمين.

c. <u>**التحقق من الرسائل.**</u>

d. البرمجيات الخبيثة *Malware*.

9. **يقدم بروتوكول المصافحة *SSL Handshake Protocol* الخدمة التالية:**

a. سلطة منح شهادات *CA*.

b. ترميز الطرود *packet encoding*.

c. التوقيع الرقمي.

d. <u>**تبادل المفاتيح *key exchange*.**</u>

10. يقدم بروتوكول السجلات *SSL Record Protocol* الخدمة التالية:

a. إدارة المفاتيح.

b. الإتاحية *availability*.

c. **<u>سلامة الرسائل.*message integrity*</u>**

d. ضبط الوصول.*access control*

11. يستخدم بروتوكول المصافحة *SSL Handshake Protocol* الشهادات الرقمية *digital certificates* لأجل:

a. توليد المفاتيح السرية

b. التوقيع الرقمي.

c. **<u>التحقق من الهوية.*authentication*</u>**

d. سرية البيانات.*confidentiality*

12. يفيد بروتوكول المصافحة *SSL Handshake Protocol* في تحقيق ما يلي:

a. إنشاء قنوات.*TCP*

b. **<u>تبادل مفاتيح التعمية.</u>**

c. تسليم طرود.*IPSec*

d. تعمية الرسائل.

13. تشترط طبقة المقابس الآمن *SSL* العمل فوق بروتوكول *TCP* لتقديم:

a. خدمة سريعة.

b. خدمة توقيع رقمي.

***c.* <u>خدمة موثوقة *reliable*.</u>**

d. خدمة تعمية.

14. يعمل البروتوكول التالي الذي ينتمي إلى طبقة المقابس الآمنة *SSL* في مستوى التطبيقات *application layer*:

a. *HTTP.*

***b.* <u>المصافحة.*handshake*</u>**

c. السجلات.*record*

d. *SMTP.*

15. إن هدف استخدام *HTTPS* هو:

a. استبدال *HTTP* ببروتوكول ويب آخر.

b. ترميز *encoding* لطرد *HTTP*.

c. ترميز لترويسة طرد *HTTP* فقط.

***d.* <u>تعمية طرد *HTTP*.</u>**

16. من أهداف البروتوكول *S / MIME*:

a. استبدال البروتوكول *SMTP*.

b. استبدال البروتوكول *HTTP*.

c. تعمية صفحات الويب.

d. **التوقيع الرقمي للبريد الإلكتروني.**

الفصل السابع: الامن الفيزيائي

1. الأنواع الرئيسية الثلاثة لمهددات الأمن الفيزيائي:

a. المهددات البيئية، البرمجيات الخبيثة، والحريق.

b. المهددات التقنية، المهددات التي يسببها الإنسان، وسرقة الحواسيب.

c. المهددات البيئية، اختراق الشبكات، التداخل الكهرطيسي.

d. **المهددات البيئية، المهددات التقنية، والمهددات التي يسببها الإنسان.**

2. أي من التالي ليس من ضمن الأمن الفيزيائي:

a. **الحماية من المخترقين *Hackers*.**

b. حماية غرفة المخدمات.

c. الحماية من التداخل الكهرطيسي.

d. الحماية من الحريق والسرقة.

3. من أهم العناصر التي تستخدم للتعافي من خروقات الامن الفيزيائي:

a. التخزين الاحتياطي للمعلومات *backup*.

b. استخدام وحدات التغذية عديمة الانقطاع *(c.UPS*

c. **تكرار التجهيزات.*redundancy***

d. تشفير البيانات المخزنة.

الفصل الثامن: إدارة الأمن - المعايير

1. السياسة الأمنية هي مجموعة من الوثائق تحدد:

a. الأهداف الأمنية للمنظومة المعلوماتية.

b. الخطوات التي تتخذها الإدارة لتحقيق الامن.

c. مواصفات العتاديات والبرمجيات.

d) **تعليمات الإدارة وقواعد أمن النظم المعلوماتية.**

2. تحدد الإدارة ثلاثة أنواع من السياسة الأمنية: *3.*

a. **سياسة أمنية عامة، سياسة أمنية تتعلق بنوع من مكونات النظام المعلوماتي، وسياسة أمنية لكل جهاز ضمن كل نوع.**

b. سياسة أمنية عامة، سياسة لكلمات المرور، وسياسة لكيفية التعامل مع الاختراق.

c. سياسة أمنية عامة، سياسة للشبكات، سياسة للاتصالات.

d. سياسة أمنية عامة، سياسة لتشفير البيانات، سياسة للأمن الفيزيائي.

4. يتضمن التخطيط للطوارئ: *5.*

a. خطط إدارة الكوارث، خطط وسياسة أمنية عامة، خطط إدارة الأمن الفيزيائي.

b. خطط التعرف على المهددات، خطط الاستجابة للحوادث *(IRPs)*، وخطط استمرارية العمل *(BCPs)*.

c. خطط الاستجابة للحوادث *(IRPs)*، تقدير الأضرار، خطط التعافي من الكوارث *(DRPs)*.

d. **خطط الاستجابة للحوادث *(IRPs)*، خطط التعافي من الكوارث *(DRPs)*، خطط استمرارية العمل *(BCPs)*.**

الفصل التاسع: الجرائم المعلوماتية

1. الجرائم المعلوماتية المستحدثة والمرتكبة عبر الشبكة المعلوماتية:

a. الترويج والاتجار بالمخدرات، اعتراض المعلومات، إرسال البريد الواغل، اعاقة الوصول للخدمة.

b. الذم والقدح، اعتراض المعلومات، إرسال البريد الواغل، اعاقة الوصول للخدمة، الاحتيال عن طريق الشبكة.

c. **إرسال البريد الواغل، اعتراض المعلومات، اعاقة الوصول للخدمة، الاحتيال عن طريق الشبكة.**

d. التهديد والابتزاز، الترويج والاتجار بالمخدرات، اعاقة الوصول للخدمة، شغل اسم موقع الكتروني.

2. الجرائم المعلوماتية المنصوص عنها في القوانين الجزائية والمرتكبة عبر الشبكة المعلوماتية:

a. التهديد والابتزاز، الذم والقدح، المساس بالآداب والاخلاق العامة، الترويج والاتجار بالمخدرات.

b. تسهيل الدعارة أو الفجور، الاساءة للمقدسات والشعائر الدينية، اعتراض المعلومات، الذم والقدح.

c. غسيل الاموال، الترويج للأعمال الإرهابية، إرسال البريد الواغل، الذم والقدح.

d. التهديد والابتزاز، الترويج والاتجار بالمخدرات، اعاقة الوصول للخدمة، الذم والقدح.

3. الدليل الرقمي هو البيانات الرقمية المخزّنة في الأجهزة الحاسوبيّة أو المنقولة عبرها والتي يمكن استخدامها في:

a. الحماية من الاختراق.

b. إثبات أو نفي جريمة.

c. التنصت على الشبكات.

d. كشف سرقة الحواسيب.

الباب السادس

الإنترنت والويب

(Internet and Web)


WWW

الفصل الأول

أساسيات الإنترنت والويب

1 - 1 - الإنترنت

- مجموعة ضخمة من الحواسب المرتبطة مع بعضها عبر شبكات الاتصالات. تختلف هذه الحواسب عن بعضها بحجمها ومصنعيها وأنظمتها.

1 - 2 - البروتوكول *TCP / IP*

- معيار يسمح بتواصل التجهيزات المختلفة مع بعضها البعض:

Transmission Control Protocol/Internet Protocol (TCP / IP)

1 - 3 - عناوين بروتوكول الانترنت *IP Addresses*

- يُعرّف الحاسب على الإنترنت بعنوان رقمي من أربعة أرقام *(IP Address)*.



My IP Address
192.168.1.1

1 - 4 - أسماء النطاق *Domain Names*

- تمّ التوافق على استخدام أسماء للحواسب عوضاً عن العناوين الرقمية.

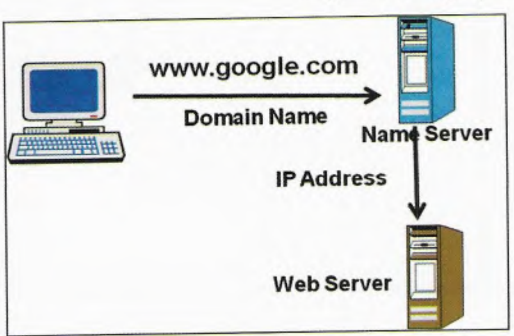
مثلاً:

movies.comedy.marxbros.com

حيث *movies* هو اسم المضيف، و *comedy* هي النطاق المحلي لـ *movies*، والتي هي جزء من النطاق *marxbros* والذي هو بدوره جزء من النطاق *com*.

1 - 5 - مخدمات الأسماء *Name Servers*

- برمجيات خاصة تقوم بتحويل أسماء النطاقات التي يكتبها المستخدم في متصفح الويب إلى العناوين الرقمية الموافقة.
- توضع هذه البرمجيات على مخدمات خاصة تُشغّل من قبل منظمات مسؤولة عن الحواسب المرتبطة بها.
- عندما يقوم مستخدم بطلب وثيقة باستخدام اسم النطاق يتمّ توجيه هذا الطلب إلى أقرب مخدم أسماء للحصول على العنوان *IP* لمخدم الوثيقة المطلوبة.



1 - 6 - الويب *World Wide Web*

- مجموعة من البرمجيات والبروتوكولات التي تمّ وضعها على حواسب الانترنت بهدف السماح للأشخاص حول العالم باستخدام الانترنت للبحث والحصول على الوثائق من أي حاسب آخر مخدم للوثائق.
- تستخدم شكلاً معيناً للوثائق يُدعى النصوص الفائقة *HyperText* والذي هو عبارة عن نص يحوي ارتباطات إلى نصوص أخرى في وثائق أخرى، مما يسمح باستعراض غير تسلسلي بالضرورة للنصوص.
- يُمكن للوثائق أن تحوي صور أو صوت أو أنواع أخرى من الوسائط، فتُدعى فائقة الوسائط *HyperMedia*.
- يُمكن أن نرى الويب بأنها مجموعة ضخمة من الوثائق المترابطة فيما بينها.

1 - 7 - متصفحات الويب *Web Browsers*

- عندما يتخاطب حاسبين في شبكة، يتصرف في معظم الأحيان أحدهما كزبون والآخر كمخدم.
- يبدأ الزبون *Client* بالتخاطب، فيطلب بشكل عام بيانات موجودة على المخدم *Server*، الذي يقوم بإرسال هذه البيانات للزبون.
- تعمل الويب باستخدام هذا المبدأ والذي ندعوه زبون-مخدم *Client / Server*.
- تعتمد الويب لتحقيق التواصل بين المستعرض والمخدم البروتوكول:

HTTP (Hyper Text Transfer Protocol)

والذي يُشكل الشكل المعياري للتخاطب بين المستعرض والمخدم.



من أشهر المتصفحات المستخدمة:

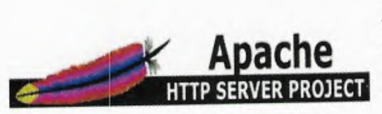
- المتصفح *Internet Explorer (IE)*.
- المتصفح *Firefox*.
- المتصفح *Google Chrome*.

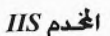
1 - 8 - مخدمات الويب

- برامج مهمتها تقديم الوثائق المطلوبة من قبل المتصفحات.
- من أشهر هذه الخدمات:

المخدم *Apache*

مخدم مجاني مفتوح المصدر يعمل على جميع أنظمة التشغيل. ويتميز بوثوقيته ودعمه للعديد من لغات البرمجة.





المخدم *IIS*

(Internet Information Services)

مخدم *Microsoft* وبالتالي فهو يعمل على النظام *Windows* فقط. وهو المنافس الأقوى للمخدم *Apache*.

IIS7
internet information services

1 - 9 - المؤثرات الأساسية للغة التأشير *XHTML*

HTML

- تتألف من مجموعة من المؤثرات والواصفات التي تُحدد كيفية إخراج وتنسيق صفحة الويب.

XHTML

- معيار معتمد لكتابة صفحات *HTML*.
- موثق جيداً.
- قواعد كتابة صارمة تفرض بنية متماسكة.
- يُمكن التحقق من توافقية أي نص مع قواعد اللغة ومعيارها باستخدام أدوات التحقق التي توفرها المنظمة *W3C*.

1 - 10 - الشكل الأساسي *Syntax*

- تُعرف عناصر *elements* اللغة باستخدام مجموعة من المؤثرات *Tags*.

- شكل المؤثر:

- Opening tag: <name>

- Closing tag: </name>

- يُشكّل مؤثر الفتح ومؤثر الإغلاق حاوية *container* للمحتوى *content*.

- لا يكون لكل المؤثرات محتوى وفي هذه الحالة يكون شكلها *<name />*

- ندعو الحاوية والمحتوى بالعنصر *element*.

- يُمكن أن يكون للمؤثر واصفات *attributes* توضع بعد اسم المؤثر:

<name attribute1="value1" attribute2="value2" ... >

- يكون للتعليق الشكل:

<!-- ... -->

- تتجاهل المتصفحات التعليقات والمؤثرات غير المفهومة والأسطر *line breaks* والفراغات المتتالية *multiple spaces* والمسافات*tabs* .

1 - 11 - البنية المعيارية لوثيقة *XHTML*

- تكون المؤثرات *<html>, <head>, <title>, <body>* مطلوبة في كل وثيقة.

- يكون المؤثر *<html>* جذر كامل الوثيقة.

- تتكون الوثيقة من رأس *head* وجسم *body*.

- يقوم المؤثر *<title>* بإظهار عنوان للوثيقة في شريط العنوان لنافذة المتصفح.

1 - 12 - أساسيات تأشير النص

الفقرات Paragraphs

```
<body>
<p>
 Mary had
 a
 little lamb, its fleece was white as snow. And
every where that
 Mary went, the lamb
was sure to go.
</p>
</body>
```

Mary had a little lamb, its fleece was white as snow. And every where that Mary went, the lamb was sure to go.

كسر السطر

```
<p>
 Mary had
 a
 little lamb, <br /> its fleece was white as snow. And
every where that
 Mary went, the lamb
was sure to go.
</p>
```

Mary had a little lamb,

its fleece was white as snow. And every where that Mary went, the lamb was sure to go.

التـرويسات *Headings*

```
<body>
    <h1> Aidan's Airplanes (h1) </h1>
    <h2> The best in used airplanes (h2) </h2>
    <h3> "We've got them by the hangarful" (h3) </h3>
    <h4> We're the guys to see for a good used airplane (h4) </h4>
    <h5> We offer great prices on great planes (h5) </h5>
    <h6> No returns, no guarantees, no refunds,
      all sales are final! (h6) </h6>
</body>
```

Aidan's Airplanes (h1)

The best in used airplanes (h2)

"We've got them by the hangarful" (h3)

We're the guys to see for a good used airplane (h4)

We offer great prices on great planes (h5)

No returns, no guarantees, no refunds, all sales are final! (h6)

الخطوط *Fonts*

- *<b>* خـط غامق *Boldface*
- *<i>* خـط مائل *Italics*
- *<big>* خـط أكـبر *Larger*

- <small> خط أصغر *Smaller*
- <tt> خط واحد *Monospace*
- <sup> خط أعلى *Superscript*
- <sub> خط أدنى *Subscript*

```
<p>
  The <big> sleet <big> in <big> <i> Crete
  </i><br /> lies </big> completely </big>
  in </big> the street
 </p>
 <p>
  <tt>
  Monospace <big> font </big>
  </tt>
 <p>
 x<sub>2</sub><sup>3</sup> + y<sub>1</sub><sup>2</sup>
 </p>
```

The sleet in *Crete*
lies completely in the street

`Monospace font`

$x_2^3 + y_1^2$

الصور *Images* : المؤثر <img>

```
<body>
  <h1> Aidan's Airplanes </h1>
  <h2> The best in used airplanes </h2>
   <img src = "images/plane.png"  alt = "Picture of a Cessna 210"
                        width="160" height="120" />
   <br />
   Buy this fine airplane today at a remarkably low price
</body>
```

Aidan's Airplanes

The best in used airplanes



Buy this fine airplane today at a remarkably low price

الروابط *Hypertext Links* : المؤثر <a>

```
<body>
  <h1> Aidan's Airplanes </h1>
  <h2> The best in used airplanes </h2>
   <a href = "6.images.html">
    Information on the Cessna 210 </a>
</body>
```

Aidan's Airplanes

The best in used airplanes

Information on the Cessna 210

عند النقر على الرابط سيتم فتح الملف الموافق:

Aidan's Airplanes

The best in used airplanes



Buy this fine airplane today at a remarkably low price

- تُستخدم الواصفة *id* لإعطاء معرف هدف:

```
<h1 id = "baskets"> Baskets </h1>
```

يُمكن استخدام هذا المعرف في الرابط (يجب سبقه بالحرف #):

```
<a href = "#baskets">  What about baskets? </a>
```

- إذا كان المعرف في وثيقة أخرى فيجب وضع اسم الوثيقة:

```
<a href = "myAd.html#baskets"> Baskets </a>
```

- يُمكن أن يكون الرابط صورة:

```
<a href = "6.images.html">
  <img src = "images/Plane.png"
    alt = "Small picture of an airplane " />
</a>
```

الجداول

يتألف الجدول من مصفوفة من الخلايا. يُمكن أن يكون لكل منها محتوى. يُمكن للخلايا أن تحوي أي عنصر.

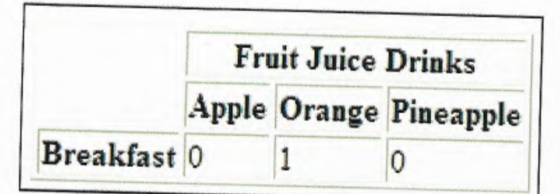
```
<body>
<table border = "border">
   <caption> Fruit Juice Drinks </caption>
   <tr>
     <th> </th>
     <th> Apple </th>
     <th> Orange </th>
     <th> Pineapple </th>
   </tr>
       <tr>
        <th> Breakfast </th>
        <td> 0 </td>
        <td> 1 </td>
        <td> 0 </td>
       </tr>
       <tr>
        <th> Lunch </th>
        <td> 1 </td>
        <td> 0 </td>
        <td> 0 </td>
       </tr>
       <tr>
        <th> Dinner </th>
        <td> 0 </td>
        <td> 0 </td>
        <td> 1 </td>
       </tr>
</table>
</body>
```

	Fruit Juice Drinks		
	Apple	**Orange**	**Pineapple**
Breakfast	0	1	0
Lunch	1	0	0
Dinner	0	0	1

امتداد الخلايا

```
<body>
<table border = "border">
    <tr>
     <td rowspan = "2"> </td>
     <th colspan = "3"> Fruit Juice Drinks </th>
    </tr>
    <tr>
     <th> Apple </th>
     <th> Orange </th>
     <th> Pineapple </th>
    </tr>
          <tr>
           <th> Breakfast </th>
           <td> 0 </td>
           <td> 1 </td>
           <td> 0 </td>
          </tr>

 </table>
 </body>
```



	Fruit Juice Drinks		
	Apple	Orange	Pineapple
Breakfast	0	1	0

المحاذاة

```
<body>
<table border = "border">
  <caption> The align and valign attributes </caption>
  <tr align = "center">
    <th> </th>
    <th> Column Label </th>
    <th> Another One </th>
    <th> Still Another One </th>
  </tr>
  <tr>
    <th> align </th>
    <td align = "left"> Left </td>
    <td align = "center"> Center </td>
    <td align = "right"> Right </td>
  </tr>
  <tr>
    <th> <br /> valign <br /> <br /> </th>
    <td> Default </td>
    <td valign = "top"> Top </td>
    <td valign = "bottom"> Bottom </td>
  </tr>
</table>
</body>
```

The align and valign attributes			
	Column Label	**Another One**	**Still Another One**
align	Left	Center	Right
valign	Default	Top	Bottom

1 - 13 - النماذج *Forms*

- تُستخدم النماذج لتحصيل مجموعة بيانات من المستخدم. ومن ثم إرسال هذه البيانات من المتصفح إلى الخدم.

المؤثر *<form>*

- توضع جميع مكونات النموذج ضمن هذا المؤثر.
- تحدّد الواصفة *action* عنوان التطبيق الذي ستُرسل بيانات النموذج إليه.
- تحدّد الواصفة *method* طريقة إرسال البيانات:

- القيمة *get* (وهي القيمة الإفتراضية): تُرسل البيانات في هذه الحالة في سلسلة محرفية تُضاف إلى المحدد *URL*.

- القيمة *post* تُرسل البيانات في هذه الحالة عبر أغراض خاصة إلى الخدم.

المؤثر <input>

- تحدد الواصفة *type* نوع الكائن المطلوب والتي تأخذ إحدى القيم التالية:
  - القيمة *text* لإنشاء صندوق نص.
  - القيمة *password* لإنشاء صندوق كلمة سر.
  - القيمة *checkbox* لإنشاء صندوق تحقق.
  - القيمة *radio* لإنشاء زر خيار.
  - القيمة *submit* لإنشاء زر إرسال.
  - القيمة *reset* لإنشاء زر إعادة.

صندوق النص *textbox*

- يكون الحجم الإفتراضي *20* محرف.
- يُمكن تحديد حجم معين باستخدام الواصفة *size*.
- في حال كتابة محارف أكثر من الحجم المحدد يظهر تلقائياً شريط إنزلاق.
- يُمكن تحديد عدد محارف أعظمي لا يُمكن تجاوزه باستخدام الواصفة *maxlength*.
- يجب إعطاء اسم لصندوق النص باستخدام الواصفة *name*.

صندوق كلمة السر *password*

- يُماثل صندوق النص في واصفاته. إلا أنه يُظهر نجوم عوضاً عن المحارف المدخلة من قبل المستخدم.

صندوق التحقق *checkbox*

- يتم إعطاء اسم لصندوق التحقق باستخدام الواصفة *name*.
- يتم وضع الواصفة *value* لتحديد قيمة لصندوق التحقق.
- يكون صندوق التحقق إما محدداً أم لا.
- يُمكن استخدام الواصفة *"checked="checked* لجعل صندوق التحقق محدداً.

زر الخيار *radio button*

- لا يُمكن في مجموعة مترابطة من أزرار الخيار تحديد سوى زر واحد.
- يجب إعطاء نفس الاسم لكل أزرار الخيار في مجموعة مترابطة.
- تُحدد الواصفة *value* قيمة لزر الخيار.
- يُمكن استخدام الواصفة *"checked="checked* لجعل زر خيار محدداً.

زر الإرسال *submit button*

- يقوم هذا الزر بحزم بيانات النموذج وإرسالها إلى الوجهة المحددة بالواصفة *action* للنموذج وبالطريقة المحددة بالواصفة *method*.

زر الإعادة *reset button*

- يُعيد هذا الزر عناصر التحكم في النموذج إلى قيمها الابتدائية.

مثال:

```
<body>
<form action = "Server.html">
<p>Fiance<input type = "text" name ="Fiance" size="12" /></p>
<p>Salary <input type = "password" name ="Salary" size ="12"
/> </p>
<p>
Own
<input type = "checkbox" name ="CO1" value="House"
                checked = "checked"/> House
<input type = "checkbox" name ="CO2" value="Car"/> Car
<input type = "checkbox" name ="CO3" value="Gold"/> Gold
</p>
<p>
Age
<input type = "radio"  name ="age" value = "under20"
                checked = "checked"/> 0- 19
<input  type ="radio"  name ="age" value = "20- 35"/> 20- 35
<input  type ="radio"  name ="age" value = "36- 50"/> 36- 50
<input  type ="radio"  name ="age" value = "over50"/> Over 50
</p>
<input  type ="reset"  value = "Reset Form"/>
<input  type ="submit" value = "Submit Form"/>
</form>
</body>
</html>
```

Fiance Bassel

Salary •••••

Own ☑ House ☐ Car ☐ Gold

Age ○ 0-19 ○ 20-35 ◉ 36-50 ○ Over 50

Reset Form | Submit Form

لاحظ أنه في هذا المثال نُرسل البيانات المدخلة إلى صفحة أخرى *Server.html* باستخدام الطريقة الافتراضية *get*:

Server.html?Fiance=Bassel&Salary=15000&CO1=House&age=36-50

المؤثر *<select>*

- يُستخدم المؤثر *<select>* لإنشاء قائمة.
- يُستخدم المؤثر *<option>* لكل خيار في القائمة.

- يُمكّن السلوك الإفتراضي للقائمة من اختيار عنصر وحيد.
- إذا أردنا تحقيق إمكانية خيارات متعددة فيجب وضع الواصفة *"multiple="multiple*.
- يُمكن استخدام الواصفة *size* لتحديد عدد العناصر الظاهرة.
- يُمكن جعل خيار محدد افتراضياً بوضع الواصفة *"selected="selected* للمؤثر *<option>*.

مثال:

```
<form action = "">
<p>
  <select name = "groceries" >
   <option> milk </option>
   <option> bread  </option>
   <option selected="selected"> eggs </option>
   <option> cheese  </option>
  </select>
</p>
```



المؤثر <textarea>

- يُستخدم المؤثر <textarea> لإنشاء صندوق نص متعدد الأسطر.

```
<form action = "">
 <p>
 Please provide your employment aspirations
 </p>
 <p>
 <textarea name = "aspirations"  rows = "4" cols="40">
 (Be brief and concise)
 </textarea>
 </p>
</form>
```

Please provide your employment aspirations

(Be brief and concise)

1 - 14 - أسلوب الصفحات المتتالي

- يوفر أسلوب الصفحات المتتالي *Cascade Style Sheet (CSS)* طرق التحكم بمظهر الوثائق.
- يُمكن تعريف أسلوب الصفحات وفق ثلاثة مستويات متدرجة. يُهيمن المستوى الأدنى على المستوى الأعلى.

1 - 15 - مستويات أسلوب الصفحات

- **الأسلوب الفوري** *Inline*: يُحدد الأسلوب لظهور معين لمؤثر. يظهر هذا الأسلوب في المؤثر نفسه.

- **أسلوب الوثيقة** *Document-level style sheets*: يُحدد الأسلوب لكامل الوثيقة. يظهر هذا الأسلوب في رأس *head* الوثيقة.

- **الأسلوب الخارجي** *External style sheets*: يُمكن تطبيقه على مجموعة من الوثائق. يُكتب هذا الأسلوب في ملف نصي له اللاحقة *css*.

```
<link rel = "stylesheet"  type = "text/css"

href = "http://www.wherever.org/termpaper.css">

</link>
```

1 - 16 - تنسيق تحديد الأسلوب

- **الأسلوب الفوري *Inline*:** يكون الأسلوب قيمة للواصفة *style*، ويأخذ الشكل العام التالي:

```
style = "property_1: value_1;
      property_2: value_2;
      ...
      property_n: value_n"
```

مثال:

```
<p style= "font-size: 14pt;
      font-style: italic;
      font-family: 'Times New Roman';">
   If a job is worth doing, it's worth doing right.
</p>
```

- **أسلوب الوثيقة *Document-level style sheets*:** يُحدّد الأسلوب كمجموعة من القواعد ضمن المؤثر <style>.

```
<style type = «text/css»>
      rule list
</style>
```

حيث تأخذ القاعدة الشكل:

```
selector {property_1: value_1;
      property_2: value_2;
      ...
      property_n: value_n}
```

- **الأسلوب الخارجي *External style sheets*:** يُكتب هذا الأسلوب في ملف نصي يحوي مجموعة من القواعد من الشكل السابق.

1 - 17 - أشكال المحددات

شكل المحدد البسيط *Simple Selectors*

```
h1 {font-size: 24pt;}
h2, h3 {font-size: 20pt;}
```

التحديد السياقي *Contextual Selectors*

```
body b i {font-size: 30pt; }
```

مثال:

```
<html xmlns = "http://www.w3.org/1999/xhtml">
<head> <title> Selector </title>
<style type="text/css" >
h1 {font-size: 24pt;}
h2, h3 {font-size: 20pt;}
body b i {font-size: 30pt; }
</style>
</head>
<body>
<h1> This is heading 1</h1>
<h2> This is heading 2</h2>
<h3> This is heading 3</h3>
<p>
This is to test <b> <i> contextual </i></b> selector
</p>
</body>
</html>
```

This is heading 1

This is heading 2

This is heading 3

This is to test ***contextual*** selector

محددات الصف *Class Selectors*

```
{;p.narrow {font-size=14
{;p.wide {font-size=20
```

مثال:

```
<html xmlns = "http://www.w3.org/1999/xhtml">
<head> <title> Selector </title>
<style type="text/css" >
p.narrow {font-size:14;}
p.wide {font-size:20;}
</style>
</head>
<body>
<p class="narrow">
This is a narrow paragraph !
</p>
<p class="wide">
This is a wide paragraph !
</p>
</body>
</html>
```

```
This is a narrow paragraph !

This is a wide paragraph !
```

المحددات العامة *Generic Selectors*

يُمكن تعريف صفوف عامة بهدف تطبيق نفس الأسلوب على عدة مؤثرات.

```
.really-big {font-size:30; }
```

تُستخدم الواصفة *class* مع مؤثرات مختلفة:

```
<h1 class = "really-big"> ... </h1>
...
<p class = "really-big"> ... </p>
```

محددات المعرفات *id Selectors*

يسمح بتطبيق أسلوب على عنصر واحد معين (له نفس المعرف). ويأخذ الشكل:

```
#specific-id {property-value list }
```

فمثلاً إذا تم تعريف:

```
#section14 {font-size: 20}
```

فسيتم تطبيق هذا الأسلوب على :

```
<h2 id="section14"> Special Heading </h2>
```

1 - 18 - أشكال خاصية-قيمة *Property-Value*

- يوجد حوالي 60 خاصية تتوزع على 7 فئات:
  - الخطوط *Fonts*
  - القوائم *Lists*
  - محاذاة النص *Text Alignment*
  - الهوامش *Margins*
  - الألوان *Colors*
  - الخلفيات *Backgrounds*
  - الحدود *Borders*
- يُمكن أن تكون قيم الخصائص كلمات مفتاحية مثل *large, medium, small*.

مثلاً:

font-size:large

أو قيم رقمية مع الوحدات التالية:

- *px - pixels*
- *in - inches*
- *cm - centimeters*
- *mm - millimeters*
- *pt - points*
- *pc - picas (12 points)*
- *em - height of the letter 'm'*
- *ex-height - height of the letter 'x'*

مثلاً:

font-size:14pt

- يُمكن استخدام النسبة المئوية لتحديد أن القيمة الحالية هي نسبة مئوية من القيمة السابقة. فمثلاً إذا وضعنا:

font-size:75%

يُصبح حجم الخط *75%* من القيمة السابقة لحجم الخط.

- تأخذ قيم محددات المصدر الشكل:

url(. . .)

مثلاً:

body {background-image: url(photo.png);}

- يُمكن تحديد قيمة لون:

- إما باستخدام اسم اللون:

Color:white

- أو بوضع قيمة اللون بالنظام السداسي عشر:

Color:#FFFFFF

- أو باستخدام الوظيفة *rgb*:

Color:rgb(255, 255, 255)

1 - 19 - خصائص الخط Font Properties

اسم الخط *font-family*

يستخدم المتصفح أول خط يدعمه من القائمة:

font-family: Arial, Helvetica, Courier

إذا كان اسم الخط أكثر من كلمة فيجب استخدام الحرف ' :

font-family: 'Times New Roman'

حجم الخط font-size

- **رقمية:** حيث تكون القيمة إما رقم يُحدد حجم الخط أو نسبة مئوية من الخط السابق:

font-size:10pt

font-size:75%

- **كلمة مفتاحية:** *xx-small, x-small, small, medium, large, x-large, xxlarge.*

font-size:medium

نمط الخط font-style

تكون قيمها *italic, normal.*

وزن الخط font-weight

- **كلمة مفتاحية:** *bolder, lighter, bold, normal.*

font-weight:bolder

- **رقمية** حيث تكون القيمة من مضاعفات *100* ومحصورة بين *100* و*900*. حيث القيمة *400* تكافئ *normal* والقيمة *700* تكافئ *bold*.

font-weight:800

الخاصية font

يُمكن استخدام الخاصية *font* لوضع قيم للخصائص السابقة وبالترتيب التالي: النمط، الوزن، الحجم، أسماء الخطوط:

font: italic bolder 14pt Arial Helvetica

زخرفة الخط *font-decoration*

تكون قيمها:

line-through, overline, underline.

أمثلة

```
<html xmlns = "http://www.w3.org/1999/xhtml">
 <head> <title> Font properties </title>
  <style type = "text/css">
   p.big {font-size: 14pt;
        font-style: italic;
        font-family: 'Times New Roman';}
   p.small {font: 10pt bold 'Courier New';}
   h2 {font-family: 'Times New Roman';
      font-size: 24pt; font-weight: bold}
   h3 {font-family: 'Courier New'; font-size: 18pt}
  </style>
 </head>
 <body>
  <p class = "big">
   If a job is worth doing, it's worth doing right.
  </p>
  <p class = "small">
   Two wrongs don't make a right, but they certainly
   can get you in a lot of trouble.
  </p>
  <h2> Chapter 1 Introduction </h2>
  <h3> 1.1 The Basics of Computer Networks </h3>
 </body> </ html>
```

If a job is worth doing, it's worth doing right.

Two wrongs don't make a right, but they certainly can get you in a lot of trouble.

Chapter 1 Introduction

1.1 The Basics of Computer Networks

1 - 20 - استخدام ملف خارجي css

```
<html xmlns = "http://www.w3.org/1999/xhtml">
<head> <title> External style sheets </title>
  <link rel = "stylesheet"  type = "text/css"
     href = "styles.css" />
 </head>
 <body>
  <p class = "big">
   If a job is worth doing, it's worth doing right.
  </p>
  <p class = "small">
   Two wrongs don't make a right, but they certainly
   can get you in a lot of trouble.
  </p>
  <h2> Chapter 1 Introduction </h2>
  <h3> 1.1 The Basics of Computer Networks </h3>
 </body>
</html>
```

1 - 21 - محتوى الملف styles.css:

```
p.big {font-size: 14pt;
      font-style: italic;
      font-family: ‹Times New Roman›;
      }
p.small {font: 10pt bold 'Courier New';}
h2 {font-family: 'Times New Roman';
     font-size: 24pt; font-weight: bold}
h3 {font-family: 'Courier New';
     font-size: 18pt}
```

If a job is worth doing, it's worth doing right.

Two wrongs don't make a right, but they certainly can get you in a lot of trouble.

Chapter 1 Introduction

1.1 The Basics of Computer Networks

الفصل الثاني

أساسيات JavaScript

2 - 1 - كتابة خطاطات JavaScript

- يُمكن تضمين الخططات مباشرة في ملف XHTML:

```
<script type="text/javaScript">
    -- JavaScript script –
</script>
```

- أو وضع الخططات في ملف نصي مستقل وتضمينها باستخدام:

```
<script type="text/javaScript"
     src="myScript.js">
</script>
```

2 - 2 - المُعرّفات Identifiers

- يجب أن تبدأ بحرف أو تحت السطر (_) أو إشارة الدولار $.
- لايوجد حد لطول المعرف.
- اللغة حساسة لحالة الأحرف case sensitive.

2 - 3 - الكلمات المفتاحية

break, case, catch, continue, default, delete, do, else, finally, for, function, if, in, instanceof, new, return, switch, this, throw, try, typeof, var, void, while, with.

2 - 4 - التعليقات

- يُمكن وضع التعليقات على سطر باستخدام // أو على أكثر من سطر باستخدام /* . . . . */.

2 - 5 - التعليمات

- يُستحسن وضع كل تعليمة على سطر وإنهائها بوضع فاصلة منقوطة ;.

2 - 6 - الأنماط والعمليات والتعابير

الأنماط الأساسية

Number, String, Boolean, Undefined, Null.

الأرقام

تُخزن الأرقام باستخدام الفاصلة العائمة مع دقة مضاعفة.

72, 7.2, .72, 72., 7E2, 7e2, .7e2, 7.e2, 7.2E-2

السلاسل النصية

تُحاط السلاسل النصية إما بإشارة تنصيص واحدة (') أو بإشارتي تنصيص(").

الأنماط الأساسية الأخرى

- النمط ***Boolean***: القيمتين *true* و*false* فقط.
- النمط ***Null***: قيمة وحيدة هي الكلمة المفتاحية *null*.
- النمط ***Undefined***: قيمة وحيدة هي *undefined*.

مثال:

```
var a;
    var b = 10;
    b = b + a;
    document.write("a: ", a, "<br />");
    document.write("b: ", b, "<br />");
```

النتيجة:

```
a: undefined
b: NaN
```

2 - 7 - التصريح عن المتغيرات

يُمكن التصريح عن متغير بشكل ضمني وذلك بإسناد قيمة له:

```
a= 10;
```

أو بشكل صريح باستخدام الكلمة المفتاحية var:

```
var a,
```

2 - 8 - العمليات الرقمية

++, --, +, -, *, /, %

مثال:

```
var a = 2,
    b = 4,
    c,
    d;
  c = 3 + a * b;
  // * is first, so c is now 11 (not 24)
  d = b / a / 2;
  // / association left, so d is now 1 (not 4)
```

الغرض Math

يوفر الغرض Math مجموعة من الطرق مثل:

floor, round, max, min, cos, sin, . . .

مثال:

Math.sin(x)

الغرض Number

يوفر الغرض *Number* مجموعة من الخصائص ذات القيم الثابتة الرقمية:

MAX_VALUE, MIN_VALUE, NaN,

POSITIVE_INFINITY, NEGATIVE_INFINITY, PI.

- تُعيد عملية جبرية مع فيضان *overflow* القيمة *NaN*.
- تُستخدم الدالة *isNaN( )* لاختبار أن متغير له القيمة *NaN*.
- للغرض *Number* الطريقة *toString( )* لإرجاع الرقم كسلسلة نصية.

جمع السلاسل النصية

تُستخدم إشارة الجمع + لجمع السلاسل النصية:

```
var x = "Hello";
    x = x + " World";
    // x now is Hello World
```

2 - 9 - تحويل الأنماط الضمني

- إذا كانت العملية عملية جمع بين رقم وسلسلة نصية يتمّ تحويل الرقم إلى سلسلة نصية.
- إذا كانت العملية عملية حسابية (غير الجمع) يتمّ تحويل السلسلة النصية إلى رقم.

- إذا فشلت عملية تحويل السلسلة النصية إلى رقم تُعاد القيمة *NaN*.

مثال:

```
var x, y, z, t;
    x = " August"+ 2007;
    // x now is August 2007
    y = 2007 + " August";
    // y now is 2007 August
    z = 7 * "3";
    // z now is 21
    t = "lo" * 3;
    // t now is NaN
```

2 - *10* - تحويل الأنماط الصريح

يُمكن طلب التحويل بين الأنماط بشكل صريح كما يلي:

- يُستخدم الباني *String* للحصول على سلسلة نصية.
- يُستخدم الباني *Number* للحصول على رقم.
- تُستخدم الطريقة *toString()* على رقم لتحويله إلى سلسلة نصية.
- يُمكن استخدام الدالة *parseInt* لتحويل سلسلة نصية إلى رقم صحيح.
- يُمكن استخدام الدالة *parseFloat* لتحويل سلسلة نصية إلى رقم عشري.

مثال:

```
var str1 = String(33.33);
    // str1 now is "33.33"
    var num1 = 6.6;
    var str2 = num1.toString();
    //str2 now is "6.6"
    var num2 = Number(str1);
    // num2 now is 33.33
    var num3 = str1 - 0;
    // num3 now is 33.33
    var num4 = parseInt(str1);
    // num4 now is 33
    var num5 = parseFloat(str1);
    // num5 now is 33.33
```

2 - 11 - خصائص وطرق السلاسل *String*

- خاصية واحدة هي *length* وتُعطي عدد الأحرف في سلسلة نصية.

- مجموعة من الطرق أهمها:

- *charAt(number)*
- *indexOf(One-character)*
- *substring(number1, number2)*
- *toLowerCase()*
- *toUpperCase()*

مثال:

```
var str="George";

str.length is 6

str.charAt(2) is 'o'

str.indexOf('r') is 3

str.substring(2, 4) is 'or'

str.toLowerCase() is 'george'
```

الطريقة *typeof*

تُعيد *typeof* نمط متغير:

```
typeof("George") is string

typeof(33) is number

typeof(true) is Boolean

var a; typeof(a) is undefined

typeof(b) is undefined (b is a not defined var)
```

الإسناد

```
a = a + 7;

a += 7;
```

الغرض *Date*

```
var today = new Date();
```

مجموعة من الطرق

toLocaleString, getDate, getMonth, getDay, getFullYear, getTime, getHours, getMinutes, getSeconds, getMilliseonds.

الإدخال والإظهار

تُستخدم الطريقة *write* للغرض *document* بشكل أساسي لإنشاء خرج:

```
var a = 25;
    document.write("The result is : <b> ", a, "</b> <br />");
```

```
The result is : 25
```

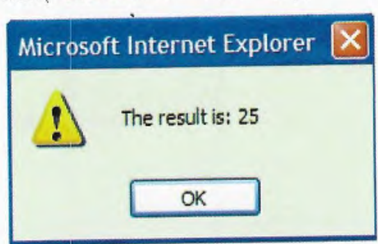
لاحظ أن معامل الطريقة *write* يُمكن أن يحوي أي مؤثر *XHTML*.

الغرض *window*

يوفر الغرض *window* ثلاثة طرق لإنشاء صناديق حوار:

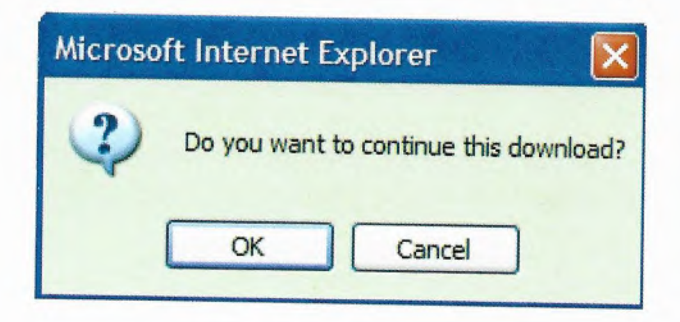
- تقوم الطريقة *alert* بفتح نافذة حوارية وإظهار محتوى معاملها فيها.

```
var a = 25;
alert("The result is: " + a + "\n");
```



- تقوم الطريقة *confirm* بفتح نافذة حوارية مع زري *OK* و *Cancel*. تكون القيمة المرجعة *true* في حال نقر المستخدم على الزر *OK*، و *false* في حال نقره على الزر *Cancel*.

```
var question =confirm("Do you want to continue this download? ");
```



- تقوم الطريقة *prompt* بإظهار نافذة حوارية تحوي صندوق نص للكتابة فيه. وتكون القيمة المرجعة هي محتوى هذا النص إذا نقر المستخدم على الزر *OK*، و *null* إذا نقر المستخدم على الزر *Cancel*.

```
a = prompt("What is your name ?", "");
```

Explorer User Prompt

Script Prompt:

What is your name ?

OK

Cancel

مثال:

```
<head>
  <title> Real roots of a quadratic equation </title>
 </head>
 <body>
  <script type = "text/javascript">
   <!--
   var question=true;
   var a, b, c, root_part, denom, root1, root2;
   while (question)
   {
   // Get the coefficients of the equation from the user
   a = prompt("What is the value of 'a'? \n", "1");
   b = prompt("What is the value of 'b'? \n", "");
   c = prompt("What is the value of 'c'? \n", "");
   // Compute the square root and denominator of the result
   root_part = Math.sqrt(b * b - 4.0 * a * c);
   denom = 2.0 * a;
   // Compute and display the two roots
   root1 = (-b + root_part) / denom;
   root2 = (-b - root_part) / denom;
   if (isNaN(root1))
   {
    alert("No real roots !");
    question=confirm("Try another equation?");
   }
   else
   {
    document.write("The first root is: ", root1, "<br />");
    document.write("The second root is: ", root2, "<br />");
    question=false;
   }
   }
   // -->
  </script>
 </body>
</html>
```

تعليمات التحكم

- تُشابه تعليمات *JavaScript* تعليمات لغة *C++* و *Java*.
- تكون التعليمات المركبة تسلسل من التعليمات المحاطة بـ *{ }*.

التعابير المنطقية

تكون نتيجة تقويم تعبير منطقي القيمة *true* أو القيمة *false*.

القيم الأساسية *Primitive values*

- إذا كانت القيمة رقمية فهي تُعتبر *true* مالم تكن مساوية للصفر.
- إذا كانت القيمة نصية تُعتبر *true* مالم تكن فارغة *""* أو *"0"*.

التعابير العلائقية *Relational Expressions*

- تستخدم العلاقات == , != , > , < , >= , <=
- في حال كون المعاملات من أنماط مختلفة يتمّ إجراء التحويل الضمني.

التعابير المركبة *Compound Expressions*

- يُمكن إنشاء تعابير مركبة من التعابير السابقة باستخدام العمليات المنطقية: *&& (And), ||(Or), !(Not)*.

2 - 12 - تعليمات الاختيار *Selection Statements*

التعليمة الشرطية *if*

مثال:

```
if (a > b)
    document.write("a is greater than b <br />");
  else {
    a = b;
    document.write(" a was not greater than b <br /> ",
           "Now they are equal <br /> ");}
```

التعليمة *switch*

```
switch (expression) {
    case value_1:
      // value_1 statements
    case value_2:
      // value_2 statements
    ...
      [default:
      // default statements]
}
```

مثال:

```
<html xmlns = "http://www.w3.org/1999/xhtml">
 <head> <title> A switch statement </title>
 </head>
 <body>
  <script type = "text/javascript">
   <!--
   var bordersize;
   bordersize = prompt("Select a table border size \n" +
              "0 , 1, 2, 4, 8 \n","1" );
   switch (bordersize) {
    case "0": document.write("<table>");
          break;
    case "1": document.write("<table border = '1'>");
          break;
    case "4": document.write("<table border = '4'>");
          break;
    case "8": document.write("<table border = '8'>");
          break;
    default:  document.write("Error - invalid choice: ",
                 bordersize, "<br />");
  }

  document.write("<caption> 2003 NFL Divisional",
          " Winners </caption>");
```

```
document.write("<tr>",
        "<th />",
        "<th> American Conference </th>",
        "<th> National Conference </th>",
        "</tr>",
        "<tr>",
        "<th> East </th>",
        "<td> New England Patriots </td>",
        "<td> Philadelphia Eagles </td>",
        "</tr>",
        "<tr>",
        "<th> North </th>",
        "<td> Baltimore Ravens </td>",
        "<td> Green Bay Packers </td>",
        "</tr>",
        "<tr>",
        "<th> West </th>",
        "<td> Kansas City Chiefs </td>",
        "<td> St. Louis Rams </td>",
        "</tr>",
        "<tr>",
        "<th> South </th>",
        "<td> Indianapolis Colts </td>",
        "<td> Carolina Panthers </td>",
        "</tr>",
        "</table>");
 // -->
 </script>
</body>
</html>
```

يبدأ التنفيذ بالطلب من المستخدم إدخال عرض الحدود المطلوب:



ومن ثم يتمّ إظهار الجدول بالعرض المحدد:

2003 NFL Divisional Winners

	American Conference	National Conference
East	New England Patriots	Philadelphia Eagles
North	Baltimore Ravens	Green Bay Packers
West	Kansas City Chiefs	St. Louis Rams
South	Indianapolis Colts	Carolina Panthers

2 - 13 - تعليمات التكرار

التعليمة *while*

```
while ( control expression)
    {
      Statements
}
```

أو:

```
do
    {
      statements
    }
    while ( control expression)
```

التعليمة *for*

```
for (initial expression; control expression; increment expression)
   {
     statements
   }
```

2 - 14 - المصفوفات *Arrays*

يتمّ تعريف المصفوفات إما باستخدام التعليمة *new* أو بإسناد عناصر المصفوفة:

```
var myList1 = new Array(24);
var myList2 = ["bread", 99, true];
```

- يكون طول المصفوفة *length* هو فهرس آخر عنصر فيها زائد *1*.

- يُمكن تغيير طول المصفوفة بإسناد قيمة إلى الخاصية *length*:

```
myList.length = 150;
```

مثال:

```
<!DOCTYPE html >
<html xmlns = "http://www.w3.org/1999/xhtml">
 <head> <title> Name list </title>
 </head>
 <body>
  <script type = "text/javascript">
   <!--
// The original list of names
var name_list= new Array("Al", "Betty", "Kasper",
                    "Michael", "Roberto", "Zimbo");
var new_name, index, last;
// Loop to get a new name and insert it
    while (new_name = prompt("Please type a new name", "")) {
    // Loop to find the place for the new name
     last = name_list.length - 1;
       while (last >= 0 && name_list[last] > new_name) {
           name_list[last + 1] = name_list[last];
           last--;
       }
   // Insert the new name into its spot in the array
     name_list[last + 1] = new_name;
  // Display the new array
    document.write("<p><b>The new name list is:</b> ",
            "<br />");
    for (index = 0; index < name_list.length; index++)
     document.write(name_list[index], "<br />");
    document.write("</p>");
   } //** end of the outer while loop
   // -->
  </script>
 </body>
</html>
```

يبدأ البرنامج بطلب اسم جديد من المستخدم:

Explorer User Prompt

Script Prompt:

Please type a new name

OK

Cancel

Mozart

ثم يُظهر المصفوفة الجديدة بعد إضافة الاسم الجديد في مكانه الصحيح:

The new name list is:
Al
Betty
Kasper
Michael
Mozart
Roberto
Zimbo

2 - 15 - بعض طرق التعامل مع المصفوفات

- *concat* لوصل مصفوفة مع أخرى.
- *join* لتشكيل سلسلة نصية من عناصر المصفوفة مع فصلها بفاصل محدد.
- *reverse* لعكس عناصر المصفوفة.
- *slice* للحصول على جزء من المصفوفة.

مثال:

```
a = new Array("a", "b", "c", "d");
    n = new Array(1, 2, 3);
    an = a.concat(n);
    document.write("a concat n= ", an, "<br />");
    document.write("a.join(',') = ", a.join(","), "<br />");
    document.write("a.slice(1,3) =", a.slice(1, 3), "<br />");
    document.write("a.reverse() =", a.reverse(), "<br />");
```

حيث يكون الخرج:

```
a concat n= a,b,c,d,1,2,3
a.join(',') = a,b,c,d
a.slice(1,3) =b,c
a.reverse() =d,c,b,a
```

2 - 16 - الوظائف *Functions*

- يُشبه شكل الوظائف في *JavaScript* شكل الوظائف في لغة *C*.

- يتمّ التصريح عن الوظائف ضمن المؤثر *<head>*.

- لايتمّ التحقق عند استدعاء الوظيفة من نمط المعاملات الممره ولا من عددها. حيث يتم تجاهل المعاملات الزائدة. أما المعاملات الناقصة فتُعتبر *undefined*.

- يتمّ إرسال المعاملات عبر مصفوفة*arguments* يُمكن الحصول على طولها من الخاصية *length*.

مثال:

```
<html xmlns = "http://www.w3.org/1999/xhtml">
 <head> <title> Parameters </title>
  <script type = "text/javascript">
        <!--
      // Function params
      // Parameters: two named parameters
      // Returns: nothing
      function params(a, b) {
          document.write("Function params was passed ",
            arguments.length, " parameter(s) <br />");
          document.write("Parameter values are: <br />");
       for (var arg = 0; arg < arguments.length; arg++)
            document.write(arguments[arg], "<br />");
       document.write("<br />");
          }
          // -->
        </script>
      </head>
      <body>
     <script type = "text/javascript">
         params("Mozart");
         params("Mozart", "Beethoven");
         params("Mozart", "Beethoven", "Tchaikowsky");
     </script>
      </body>
     </html>
```

حيث يكون الخرج:

```
Function params was passed 1 parameter(s)
Parameter values are:
Mozart

Function params was passed 2 parameter(s)
Parameter values are:
Mozart
Beethoven

Function params was passed 3 parameter(s)
Parameter values are:
Mozart
Beethoven
Tchaikowsky
```

2 - 17 - المحارف والمحارف المترفعة

- المحارف العادية وهي المحارف التي تتطابق مع نفسها.
- المحارف المترفعة وهي المحارف التي لها معنى خاص ولاتتطابق مع نفسها:

\ | ( ) [ ] { } ^ $ * + ? .

(يمكن جعل محرف مترفع يُعامل كمحرف عادي بسبقه بـ \).

الطريقة serach

تُعيد موقع بداية النموذج في السلسلة في حال وجوده (تُفهرس الأحرف اعتباراً من الصفر)، أو 1- في حال عدم وجوده.

مثال:

```
var str = "Rabbits are furry";
    var position = str.search(/bits/);
    if (position >= 0)
       document.write("'bits' appears in position",
                         position, " <br />");
    else
       document.write(" 'bits' does not appear in str <br />");
```

- يُطابق الحرف المترفع (.) أي محرف عدا السطر الجديد. مثلاً: يُطابق النموذج /.snow/ كل من snowy, snowe, snowd.

- عند وضع مجموعة من الأحرف ضمن [ ] . فهذا يعني أن المطابقة يجب أن تتم مع أحد هذه الحروف. فمثلاً يُطابق النموذج /[oi]n/ كل من on و in.

- يُمكن استخدام الحرف (-) لتعيين مجال من القيم. فمثلاً /[a-h]/ تعني أي محرف بين a و h.

- يُمكن استخدام الحرف (^) لعكس المحارف المعينة. فمثلاً /[abc^]/ تعني أي محرف ماعدا الأحرف a, b, c.

- يوجد بعض الصفوف المعرفة مسبقاً لبعض النماذج الأكثر استخداماً:

Name	Equivalent Pattern	Matches
\d	[0-9]	a digit
\D	[^0-9]	not a digit
\w	[A-Za-z_0-9]	a word character
\W	[^A-Za-z_0-9]	not a word character
\s	[\r\t\n\f]	a whitespace character
\S	[^ \r\t\n\f]	not a whitespace character

أمثلة:

- يُطابق النموذج /\d\.\d\d/ أي رقم تليه نقطة يليها رقمين.
- يُطابق النموذج /\D\d\D/ رقم واحد.
- يُمكن في العديد من الحالات تحديد تكرار معين:

Quantifier	Meaning
{n}	exactly n repetitions
{m,}	at least m repetitions
{m, n}	at least m but not more than n repetitions

فمثلاً يُطابق النموذج /xy{4}z/ السلسلة xyyyyz.

- يُستخدم الحرف (*) لتحديد صفر أو أي عدد من التكرارات.
- يُستخدم الحرف (+) لتحديد تكرار واحد أو أكثر.
- يُستخدم الحرف (?) لتحديد صفر أو تكرار واحد.

- يُطابق النموذج /?x*y+z/ سلسلة محرفية تبدأ بعدد من x (أو ولا x) يليها تكرار أو أكثر لـ y . يليها z واحدة (أو ولا z).
- يُطابق النموذج /d+\.\d*\/ رقم أو أكثر يليه نقطة يليها عدد من الأرقام (أو ولا رقم).

مثال:

```
<!DOCTYPE html >
<!-- forms_check.html
  A function tst_phone_num is defined and tested.
  This function checks the validity of phone
  number input from a form
  -->
<html xmlns = "http://www.w3.org/1999/xhtml">
 <head> <title> Phone number tester </title>
  <script type = "text/javascript">
   <!--
/* Function tst_phone_num
  Parameter: A string
  Result: Returns true if the parameter has the form of a legal
      seven-digit phone number (3 digits, a dash, 4 digits)
  */
function tst_phone_num(num) {
    var ok = num.search(/\d{3}-\d{4}/);
    if (ok == 0)
     return true;
    else
```

```
        return false;
      } // end of function tst_phone_num
      // -->
    </script>
  </head>
  <body>
    <script type = "text/javascript">
      <!--
  var tst = tst_phone_num("4445432-");
  if (tst)
    document.write("4445432- is a legal phone number <br />");
  else
    document.write("Error in tst_phone_num <br />");
  tst = tst_phone_num("444-r432");
  if (tst)
    document.write("Error in tst_phone_num <br />");
  else
    document.write("444-r432 is not a legal phone number <br />");
  tst = tst_phone_num("441234-");
  if (tst)
    document.write("Error in tst_phone_num <br />");
  else
    document.write("441234- is not a legal phone number <br />");
  // -->
  </script>
  </body>
  </html>
```

يكون الخرج:

```
444-5432 is a legal phone number
444-r432 is not a legal phone number
44-1234 is not a legal phone number
```

2 - 18 - تحديد الموقع

يُمكن استخدام الحرف *(^)* لتحديد أن موقع النموذج يجب أن يكون بداية السلسلة. أو الحرف *($)* لتحديد أن النموذج يجب أن يكون نهاية السلسلة.

أمثلة:

- يُطابق النموذج */$gold/* السلسلة *"I like gold"* بينما لايُطابق *"golden"*.
- يُطابق النموذج */^pearl/* السلسلة *"pearls are pretty"* ولايُطابق *"My pearls are pretty"*.

2 - 19 - تعديل النماذج

يُمكن استخدام الحرف *(i)* لطلب تجاهل حالة الأحرف. فمثلاً يُطابق النموذج */ok/i* كل من *OK, Ok, ok, oK*.

الطريقة *replace*

يُمكن استخدام الطريقة *replace* لاستبدال سلسلة جزئية بأخرى. كما يُمكن استخدام الحرف *(g)* لطلب الاستبدال لكل ظهور للسلسلة الجزئية.

مثال:

```
var str = "Some rabbits are rabid";
str = str.replace(/rab/g, "tim"); .
```

تصبح *str* مساوية إلى *"Some timbits are timid"*

الطريقة *match*

تُستخدم الطريقة *match* لإرجاع مصفوفة من السلاسل الجزئية المطابقة للنموذج. **مثال:**

```
var str = "My 3 kings beat your 2 aces";
var matches = str.match(/[ab]/g);
```

تصبح قيمة *matches* مساوية إلى *b,a,a*.

الطريقة *split*

تقوم الطريقة *split* بتجزئة السلسلة إلى سلاسل جزئية.

مثال:

```
var str = "red,green,blue";
var colors = str.split(",");
```

تصبح قيمة *colors* مساوية إلى *[red, green , blue]*.

2 - 20 - التفاعل بين *JavaScript* و *HTML*

يُمكن الوصول لعناصر *HTML* من خلال تعليمات *JavaScript* بعدة طرق:

2 - 21 - نموذج كائن الوثيقة *(DOM) Document Object Model*

- تُكون العناصر *elements* في هذا النموذج أغراض *objects* ، وتكون واصفاتها *attributes* خصائص *properties*.
- يحوي الكائن *document* المصفوفة *forms[]* لكل النماذج المحتواة في الوثيقة والمحددة باستخدام المؤثر *<form>*.
- يحوي كل عنصر من المصفوفة السابقة على مصفوفة *elements[]* للوصول لجميع عناصر النموذج.
- بالتالي يُمكن الوصول لأي عنصر في الوثيقة باستخدام:

document.forms[i].elements[j]

مثال:

```
<html xmlns = "http://www.w3.org/1999/xhtml">
 <head>
  <title> Access to form elements </title>
  </head>
 <body>
  <form action = "">
  <input type="button" name="turnItOn" />
  </form>
  <script type="text/javascript" >
   document.forms[0].elements[0].value="on";
  </script>
 </body>
</html>
```

- يجب الانتباه عند استخدام هذه الطريقة إلى إضافة أو حذف عناصر مما يؤدي لتغيير فهارسها.

استخدام الطريقة getElementById

يتم الوصول إلى عنصر عن طريق معرفه (يتمّ إعطاء معرف فريد للعنصر باستخدام الواصفة *id*).

مثال:

```
<html xmlns = "http://www.w3.org/1999/xhtml">
 <head>
  <title> Access to form elements </title>
 <body>
  <form  action = "">
  <input type="button" id="turnItOn" />
  </form>
  <script "type=text/javascript" >
   document.getElementById("turnItOn").value="on";
  </script>
 </body>
</html>
```

- نقوم عادةً بإعطاء نفس الاسم لمجموعة من أزرار الخيار أو مجموعة من صناديق التحقق، وعندها يكون هذا الاسم مصفوفة يُمكن من خلالها الوصول لكل عنصر في المجموعة (بالطبع، يُمكن إعطاء معرف لكل عنصر في المجموعة إلا أن ذلك لن يكون عملياً للوصول لعناصر المجموعة). فمثلاً لو كان لدينا مجموعة من صناديق التحكم:

```
<div id = "VehicleGroup">
<input type="checkbox" checked="checked" name="vehicles" value="car" /> Car
<input type="checkbox" name="vehicles" value="truck"/> Truck
<input type="checkbox" checked="checked" name="vehicles" value="bike"/> Bike
</div>
```

فيُمكن الوصول لعناصرها كما يلي (لنحسب مثلاً عدد الصناديق المختارة):

```
var numChecked = 0;
    var dom = document.getElementById("VehicleGroup");
    for (index = 0; index < dom.vehicles.length; index++)
      if (dom.vehicles[index].checked)
        numChecked++;
```

استخدام الطريقة *getElementsByName*

يُمكن الوصول إلى مجموعة من العناصر التي لها نفس الاسم باستخدام الطريقة *getElementsByName* كما يُبين المثال التالي:

```
var numChecked = 0;
    var dom = document.getElementsByName("vehicles");
    for (index = 0; index < dom.length; index++)
      if (dom[index].checked)
        numChecked++;
```

2 - 22 - الأحداث Events

- يُخبر الحدث بأن شيء محدد قد حصل بفعل المتصفح أو بفعل المستخدم.
- ندعو معالج الحدث *Event handler* الخطاطة التي تُنفذ عند وقوع الحدث.
- ندعو عملية ربط حدث بمعالج حدث بالتسجيل *Registration*.

Tag	Tag Attribute	Event	الحدث
<a>	onblur	blur	ترك العنصر
<button>			
<input>			
<textarea>			
<select>			
<input>	onchange	change	تغيير القيمة
<textarea>			
<select>			
<a>	onclick	click	النقر
<input>			
<a>	onfocus	focus	انتقال الفارة إلى العنصر
<input>			
<textarea>			
<select>			
<body>	onload	load	التحميل
Most elements	onmousedown	mousedown	ضغط زر الفارة
Most elements	onmousemove	mousemove	تحريك الفارة
Most elements	onmouseout	mouseout	مغادرة الفارة العنصر
Most elements	onmouseover	mouseover	توضع الفارة فوق العنصر
Most elements	onmouseup	mouseup	تحرير زر الفارة
<input>	onselect	select	تحديد العنصر
<textarea>			
<form>	onsubmit	submit	الإرسال
<body>	onunload	unload	إنهاء التحميل



● يتم تسجيل معالج الحدث إما بإسناد التعليمات لواصفة المؤثر:

```
<input type="button" name="myButton" id="myButton"
    onclick="alert('You clicked my button!');" />
```

● أو بكتابة تعليمات الحدث في وظيفة ومن ثم استدعاء الوظيفة في واصفة المؤثر:

```
<input type="button" name="myButton" id="myButton"
    onclick="myHandler();" />
```

● أو بإسناد اسم الوظيفة في *JavaScript* إلى خاصية الحدث:

```
document.getElementById("myButton").onclick = myHandler;
```

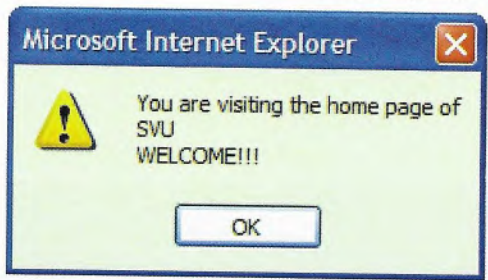
2 - 23 - معالجة أحداث جسم الوثيقة *body*

يُعتبر حدثي التحميل *load* وإنهاء التحميل *unload* من أهم أحداث المؤثر *<body>*.

● يُبين المثال التالي استخدام حدث التحميل لإظهار رسالة ترحيبية:

```
<!DOCTYPE html >
<html xmlns = "http://www.w3.org/1999/xhtml">
 <head>
<title> onLoad event handler </title>
  <script type = "text/javascript">
   <!--
// The onload event handler
   function load_greeting () {
    alert("You are visiting the home page of \n" +
         "SVU \n" +
         "WELCOME!!!");
   }
   // -->
  </script>
 </head>
 <body onload="load_greeting();">
  <p />
 </body>
</html>
```

• يُظهر المتصفح حال فتح الوثيقة:



2 - 24 - معالجة أحداث الزر

يُعتبر الحدث *onclick* الأكثر استخداماً مع زر الأمر. ويتم تسجيل الحدث عادةً إما بإسناد اسم الوظيفة إلى الواصفة *onclick*:

```
<input type="button" name="freeButton"
      id="freeButton" onclick="freeButtonHandler();">
```

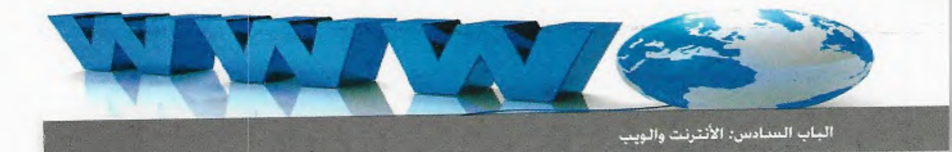
أو باستخدام:

```
document.getElementById("freeButton").onclick =
freeButtonHandler();
```

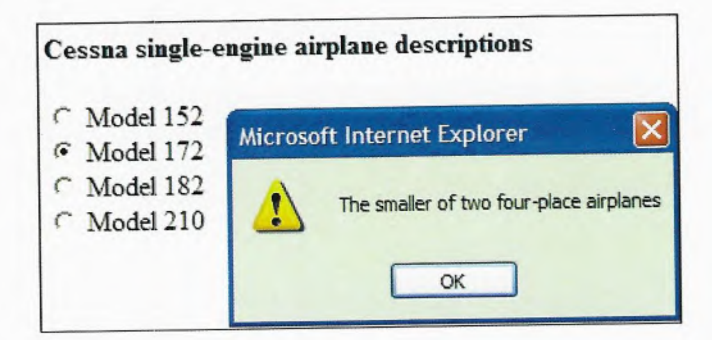
- يُبين المثال التالي استخدام الحدث *onclick* مع مجموعة من أزرار الخيار لإظهار رسالة موافقة عندما يقوم المستخدم بالنقر على زر خيار من المجموعة:

```
<html xmlns = "http://www.w3.org/1999/xhtml">
 <head>
  <title> Illustrate messages for radio buttons </title>
 <script type = "text/javascript">
  <!--
// The event handler for a radio button collection
   function planeChoice (plane) {
// Produce an alert message about the chosen airplane
    switch (plane) {
     case 152:
      alert("A small two-place airplane for flight training");
      break;
     case 172:
      alert("The smaller of two four-place airplanes");
      break;
```

```
        case 182:
         alert("The larger of two four-place airplanes");
         break;
       case 210:
         alert("A six-place high-performance airplane");
         break;
       default:
         alert("Error in JavaScript function planeChoice");
         break;
     }
    }
    // -->
   </script>
   </head>
 <body>
   <h4> Cessna single-engine airplane descriptions </h4>
   <form id = "myForm"  action = "handler">
    <p>
     <input type = "radio"  name = "planeButton"
     value = "152" onclick = "planeChoice(152)" /> Model 152
     <br />
     <input type = "radio"  name = "planeButton"
      value = "172" onclick = "planeChoice(172)" />
                          Model 172   <br />
     <input type = "radio"  name = "planeButton"
      value = "182" onclick = "planeChoice(182)" /> Model 182
     <br />
     <input type = "radio"  name = "planeButton"
      value = "210" onclick = "planeChoice(210)" /> Model 210
    </p>
   </form>
  </body>
</html>
```

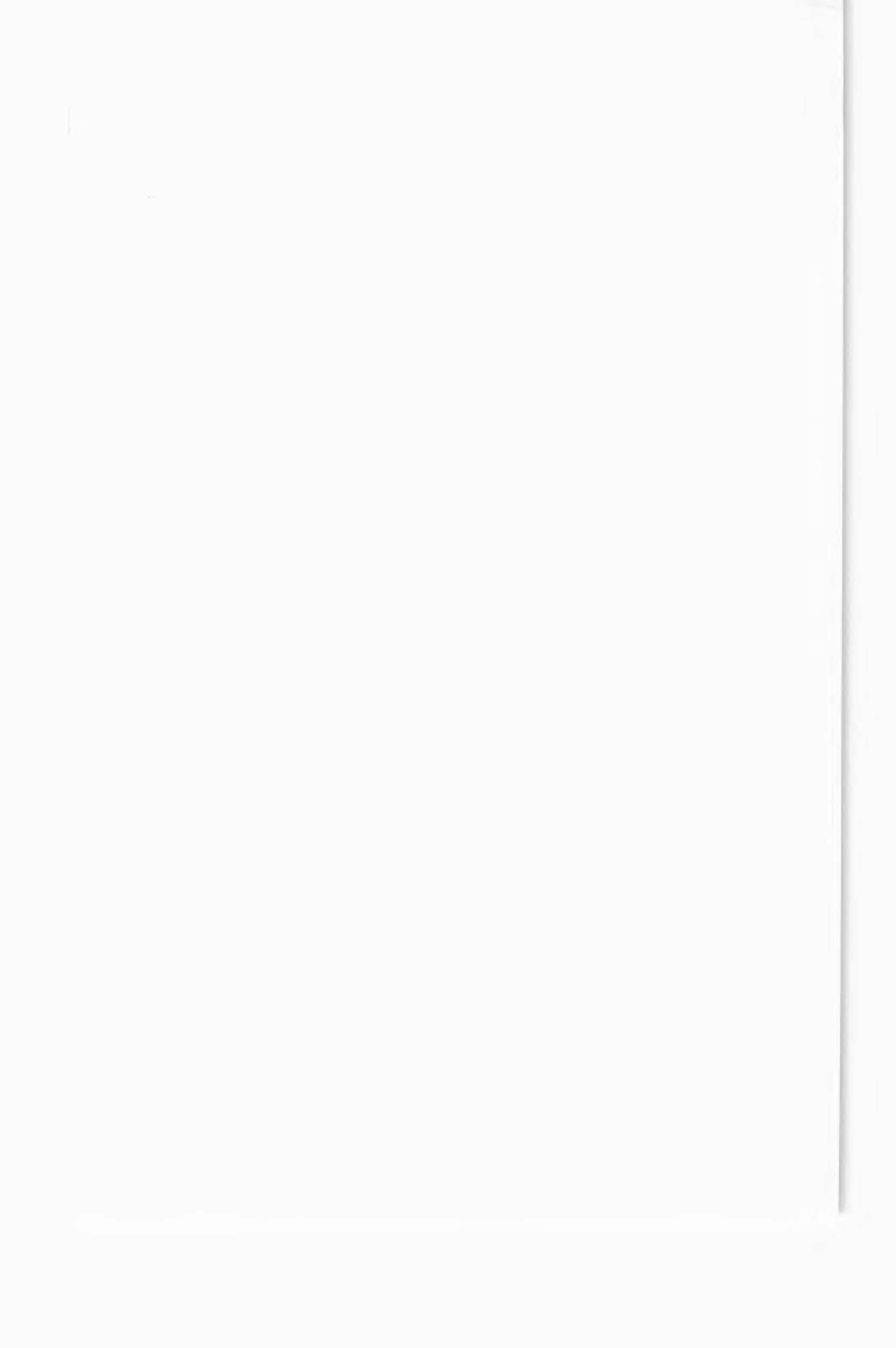


- عند النقر على أحد عناصر مجموعة أزرار الخيار تظهر الرسالة الموافقة:



الباب السابع

المصطلحات المستخدمة



حرف الألف

Simplex	اتصال بسيط
Duplex	اتصال مزدوج
Regression Testing	اختبار الانحدار (الارتداد)
Stress Testing	اختبار الإجهاد
Release Testing	اختبار الإصدار
Performance Testing	اختبار الأداء
Security Testing	اختبار الأمان
Software Testing	اختبار البرمجيات
Integration Testing	اختبار التكامل
Scenario-Based	اختبار السيناريو
Checksum	اختبار المجموع
User Testing	اختبار المستخدم
System Testing	اختبار النظام
Unit Testing	اختبار الوحدات
Alpha Testing	اختبار ألفا
Beta Testing	اختبار بيتا
White Box Testing	اختبارات الصندوق الأبيض
Black Box Testing	اختبارات الصندوق الأسود
Congestion	اختناق
Requirement Elicitation	استيضاح المتطلبات
Derivation	اشتقاق
Requirements Discovery	اكتشاف المتطلبات
Connection-Oriented	الاتصال الموجه

الاتصال غير الموجه	*Connectionless*
الاختبار القائم على النموذج	*Model-Based Testing*
الاستبيانات	*Surveys*
الاشتقاق اليساري	*Leftmost Derivation*
الاشتقاق اليميني	*Rightmost Derivation*
الانتقالات	*Transitions*
الإتاحية	*Availability*
الإجرائيات الرشيقة	*Agile Processes*
الأنماط المعمارية	*Architectural Patterns*
الأوتومات المنتهي	*Finite Automaton*
الأوتومات المنتهي الحتمي	*Deterministic Finite Automaton*
الأوتومات المنتهي اللاحتمي	*Non-Deterministic Finite Automaton*
الأوتومات ذات المكدس	*Push-Down Automaton*
الآلة ذات السجلات	*Register Machine*
الآلة ذات المكدس	*Stack Machime*
البنية الوسيطة	*Intermediate Structure*
البوابة	*Gateway*
البيانات المغلفة	*Enveloped Data*
البيانات الموقعة	*Signed Data*
البيانات الواضحة الموقعة	*Clear Signed Data*
التحقق من الرسائل	*Message Authentication*
التحقق من صحة المتطلبات	*Requirements Validation*
التحقق من هوية الطرف الآخر	*Peer Authentication*
التحقق من هوية المستخدم	*User Authentication*

التحقق من هوية مصدر البيانات	*Data Source Authentication*
التحليل الدلالي	*Semantic Analysis*
التحليل الصاعد	*Bottom-Up Parsing*
التحليل القواعدي	*Syntaxical Analysis*
التحليل المفرداتي	*Lexical Analysis*
التحليل النازل	*Top-Down Parsing*
الترابط الأمني	*SA: Security Association*
التركيب	*Composition*
التصميم المعماري	*Architectural Design*
التطوير الديناميكي	*Dynamic Development*
التطوير المتكيف	*Adaptive Development*
التطوير المعتمد على الميزات	*Feature Driven development*
التعمية التقليدية	*Conventional Encryption*
التعمية اللامتناظرة	*Asymmetric Encryption*
التعمية المتناظرة	*Symmetric Encryption*
التعمية أو التشفير	*Cryptography*
التعمية بالمفتاح السري	*SKC: Secret-Key Cryptography*
التعمية بالمفتاح العمومي	*PKC: Public-Key Cryptography*
التعميم	*Generalization*
التغليف	*Encapsulation*
التقسيم المتكافئ	*Equivalence Partitioning*
التمديدات متعددة الأغراض لبريد الإنترنت	*MIME: Multipurpose Internet Mail Extensions*
الجهة الفاعلة	*Actors*

الحالات	*States*
الحدث	*Event*
الخصوصية	*Privacy*
السرية	*Confidentiality*
السلامة	*Integrity*
الظرف الرقمي	*Digital Envelope*
العمود الفقري	*Backbone*
القواعد الصرفية	*Syntax Rules*
القياسات البرمجية	*Software Metrics*
القياسات الحيوية	*Biometrics*
اللوغاريتم المتقطع	*Discrete Logarithm*
المتطلبات	*Requirements*
المتطلبات التنظيمية	*Organizational Requirements*
المتطلبات الخارجية	*External Requirements*
المتطلبات الوظيفية	*Functional Requirements*
المتطلبات غير الوظيفية	*Non-Functional Requirements*
المختبرون	*Testers*
المخططات السلوكية	*Behavioral Diagrams*
المراجعات الفنية	*Technical Reviews*
المراقبة	*Observation*
المساءلة	*Accountability*
المسح من اليسار	*Left Scanning*
المسح من اليمين	*Right Scanning*
المشرفون	*Maintainers*

المقابلة	*Interview*
المكدس	*Stack*
المكوم	*Heap*
النحو الصرفي	*Grammar*
النشر	*Deployment*
النشر المباشر	*Direct Deployment*
النشر المتوازي	*Parallel Deployment*
النشر المرحلي	*Phased Deployment*
النماذج الهيكلية	*Structural Models*
النمذجة	*Modelling*
النموذج الإجرائي	*Process Model*
النموذج الإجرائي الموحد	*Unified Process Model*
النموذج التزايدي	*Incremental Model*
النموذج الحلزوني	*Spiral Model*
الوثوقية	*Reliability*
إدارة المتطلبات	*Requirements Management*
إدارة المخاطر	*Risk Management*
إدارة إعادة الاستخدام	*Reusability Management*
إدارة تكوين البرمجيات	*Software Configuration Management*
إدارة جودة البرمجيات	*Quality Management*
إرسال	*Transmission*
إطار	*Frame*
إطار عمل الإجرائية البرمجية	*Software Process Framework*
إعادة تشكيل	*Reassembly*

أبجدية	*Alphabet*
أخطاء المفردات	*Lexical Errors*
أصحاب المصلحة	*Stakeholders*
أعمال البناء	*Construction*
أمثَلة الرماز	*Code Optimization*
أمثلة الرماز	*Code Optimization*
أمن طبقة النقل	*TLS: Transport Layer Security*
أوتومات	*Automaton*
آلة افتراضية	*Virtual Machine*
آلة افتراضية	*Virtual Machine*

حرف الباء

برمجية خبيثة	*Malware*
برنامج مصدري	*Source Program*
برنامج هدف	*Destination Program*
بروتوكول الإنترنت الآمن	*IPSec*
بروتوكول الإنذار في *(SSL)*	*SSL Alert Protocol*
بروتوكول التحكم بالنقل	*TCP: Transmission Control Protocol*
بروتوكول الخفقان في *(SSL)*	*SSL Heartbeat Protocol*
بروتوكول السجلات في *(SSL)*	*SSL Record Protocol*
بروتوكول تشارك مفاتيح	*Key Sharing Protocol*
بروتوكول تغيير محددات التعمية في *(SSL)*	*SSL Change Cipher Spec Protocol*
بروتوكول مصافحة *(SSL)*	*SSL Handshake Protocol*
بروتوكول نقل البريد البسيط	*SMTP: Simple Mail Transfer Protocol*
بوابة	*Port*

حرف التاء

تابع تهشير	Hash Function
تبادل المفاتيح السرية	Secret Key Exchange
تحديد أولوية المتطلبات والتفاوض	Requirements Prioritization and Negotiation
تحليل المتطلبات	Requirements Analysis
تحليل صاعد	Bottom-Up Parsing
تحليل قيمة الحدود	Boundary Value Analysis
تحليل نازل	Top-Down Parsing
ترميز	Encoding
ترويسة	Header
ترويسة طرد	Packet Header
تسيير	Routing
تشبيك بيني	Interworking
تصنيف المتطلبات	Requirement Classifications
تعبير منتظم	Regular Expression
تعبير منتظم	Regular Expression
تعمية - أو تشفير	Encryption
تعميم - إذاعي	Broadcasting
تغطية العبارة	Statement Coverage
تغطية الفرع	Branch Coverage
تغطية المسار	Path Coverage
تغليف	Encapsulation
تقديم	Presentation

تقسيم	*Segmentation*
تهيئة	*Configuration*
توقيع رقمي	*Digital Signature*
توليد الرماز	*Code Generation*
توليد الرماز	*Code Generation*

حرف الجيم

جدول الرموز	*Symbol Table*
جدول الرموز	*Symbol Table*
جذر أولي	*Primitive Root*
جلسة	*Session*

حرف الحاء

حشو	*Padding*

حرف الخاء

خدمة	*Service*
خطأ صرفي	*Syntax Error*

حرف الدال

دمج تسلسلي	*Concatenation*

حرف الراء

رقم تسلسلي	*Sequence Number*
رقم عشوائي	*Random Number*
رمز البداية	*Starting Symbol*
رمز أولي	*Terminal Symbol*
رمز وسيط	*Non Terminal Symbol*

حرف الزاي

زبون	*Client*

حرف السين

سحب / اختصار	*Shift / Reduce*
سرية غير شرطية	*Unconditional Security*
سلسلة الكتل المشفرة	*CBC: Cipher Block Chaining*
سلطة منح الشهادات	*CA: Certificate Authority*
سيل مفتاحي	*Keystream*

حرف الشين

شبكة افتراضية خاصة	*VPN: Virtual Private Network*
شبكة جزئية	*Subnet*
شبكة محلية لاسلكية	*WLAN: Wireless Local Area Network*
شجرة الاشتقاق	*Derivation Tree*
شجرة الاشتقاق	*Derivation Tree*
شهادة المفتاح العمومي	*Public Key Certificate*
شهادة رقمية	*Digital Certificate*

حرف الصاد

صناديق زمنية	*Time Boxed*
صوان	*Buffer*

حرف الطاء

طبقة التطبيق	*Application Layer*
طبقة الشبكة	*Network Layer*
طبقة المقابس الآمنة	*SSL: Secure Sockets Layer*
طبقة النقل	*Transport Layer*
طرد	*Packet*
طرد	*Packet*

طرف ثالث موثوق	*TTP: Trusted Third Party*
طرق المراجعة الرسمية	*Formal Reviews*

حرف العين

علاقة الارتباط	*Association Relationship*
علاقة الوراثة	*Inheritance*
علامة	*Token*
عناصر التدفق	*Flow Objects*
عنوان إنترنت	*IP Address*
عودية يسارية	*Left Recursion*

حرف الغين

غموض	*Ambiguity*
غير ممكن حسابياً	*Computationally Infeasible*

حرف الفاء

فك التعمية أو التشفير	*Decryption*

حرف القاف

قناع الشبكة	*Net Mask*
قيمة التهشير	*Hash Value*

حرف الكاف

كتاب الترميز الإلكتروني	*ECB: Electronic Code Book*
كلمة	*Word*
كود التحقق من الرسالة	*MAC: Message Authentication Code*

حرف اللام

لاحقة	*Trailer*
لصاقة تحقق	*Authentication Tag*

لغات صورية	*Formal Languages*
لغة	*Language*
لغة النمذجة الموحدة	*Unified Modelling Language*
لغة خارج السياق	*Context-Free Language*
لغة منتظمة	*Regular Language*

حرف الميم

مالكي النظام	*System Owners*
مبدلة	*Switch*
مترجم	*Compiler*
متصفح الويب	*Web Browser*
متطفل	*Intruder*
متطلبات المستخدم	*User Requirement*
متطلبات المنتج	*Product Requirements*
متطلبات النظام	*System Requirements*
مجمع	*Hub*
مجموعة الرموز الأولى	*First*
مجموعة الرموز اللاحقة	*Follow*
محتوى الأمن المغلف	*ESP: Encapsulated Security Payload*
مختصر الرسالة	*Message Digest*
مخدم	*Server*
مخدم	*Server*
مخطط الصفوف	*Class Diagram*
مخططات التتابع	*Sequence Diagrams*

مخططات التعاون	*Collaboration Diagram*
مخططات الحالة	*State Diagrams*
مخططات النشاط	*Activity Diagrams*
مخططات حالة الاستخدام	*Use Case Diagrams*
مديري النظام	*System Managers*
مركز توزيع المفاتيح	*KDC: Key Distribution Center*
مسؤولو النظام	*System Administrators*
مشفر	*Cipher*
مشفر تسلسلي	*Stream Cipher*
مشفر كتلي	*Block Cipher*
مصدر الشهادات	*Certificate Issuer*
مضيف	*Host*
معرّف الشبكة	*Netid*
معرّف المضيف	*Hostid*
معيار التعمية المتقدم	*AES: Advanced Encryption Standard*
معيار تعمية البيانات	*DES: Data Encryption Standard*
مفتاح الجلسة	*Session Key*
مفتاح خاص	*Private Key*
مفتاح سري	*Secret Key*
مفتاح عمومي	*Public Key*
مفتاح متناظر	*Symmetric Key*
مفتاح مشترك	*Shared Key*
مفردات	*Tokens*

Socket	مقبس
Protocol Stack	مكدس البروتوكولات
TCP / IP Stack	مكدس بروتوكولات الإنترنت
Software Testing Views	مناظير اختبار البرمجيات
Work Product	منتج مرحلي
Non-Repudiation	منع الإنكار
Fault Tolerance	موارية الخطأ
Router	موجه
Support Staff	موظفي الدعم
PRNG: Pseudo-Random Number Generator	مولد الأرقام شبه العشوائية
Random Byte Generator	مولد بايتات عشوائي
SPI: Security Parameters Index	مؤشر معاملات الأمن

حرف النون

Grammar	نحو صرفي
Clean Grammar	نحو نظيف
Peer	ند
Umbrella Activities	نشاطات المظلة
Active	نشط - فاعل
Ciphertext	نص مشفر
Plaintext	نص واضح
Transport	نقل
Plan-Driven Process Models	نماذج الاجرائيات التي تعتمد على التخطيط

نماذج التفاعل	*Interaction Models*
نماذج السياق	*Context Models*
نمط الأنابيب والفلترة	*Pipe and Filter Pattern*
نمط الطبقات	*Layered Pattern*
نمط المستودع	*Repository Pattern*
نمط النفق	*Tunnel Mode*
نمط النقل	*Transport Mode*
نمط ثلاث طبقات مخدم - عميل	*Three Layers Client / Server*
نمط مخدم - عميل	*Client-Server Pattern*
نمط نموذج - عرض - تحكم	*Model-View-Controller (MVC) Pattern*
نموذج البرمجة القصوى	*Extreme Programming (XP)*
نموذج الشلال	*Waterfall Model*
نموذج أولي	*Prototype*
نموذج تدفق المعطيات	*Data Flow Model*
نموذج علاقات الكائنات	*Entity Relationship Model*

حرف الهاء

هجوم الرجل في المنتصف	*Man-in-the-Middle Attack*
هجوم القوة الغاشمة	*Brute-Force Attack*
هجوم حجب الخدمة	*DoS: Denial of Service Attack*
هندسة المتطلبات	*Requirement Engineering*

حرف الواو

واجهة تخاطبية	*Interface*
وسط	*Medium*
وسيط	*Proxy*

